Experimental Analysis of Representation Learning Systems

Vicente Ivan Sanchez Carmona

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

September 17, 2018

I, Vicente Ivan Sanchez Carmona, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Studying a subject is central to understanding its behavior and what it has learned. In this thesis, we study specific aspects of five representation learning systems for natural language processing tasks. Representation learning systems are a type of machine learning system dedicated to learn representations of data suitable for other machine learning systems, such as classifiers, to operate upon them. Thus, understanding the behavior of and the abilities learned by representation learning systems is crucial for improving the results on the tasks they are used.

The aspects on which we focus are interpretability, robustness, and abilities learned. We are interested in obtaining explanations that allow us to understand how a system makes a decision, what factors from the data and internal to the system affect its robustness, and to what extent it has learned a linguistic ability. To do so, we propose to carry out three types of analyses, namely functional, behavioral, and internal analyses which we link with work on the cognitive science, behavioral science, and neuroscience.

We present three case studies. In the first study, we provide a functional explanation of a matrix factorization system that allow us to understand how this system makes a prediction. In our second study, we investigate how robust are three systems when the input data suffers a simple transformation and how certain external and internal factors influence their behavior; these systems are trained for the task of natural language inference. Finally, our third study shows that we are able to extract hypernymy from the word embeddings of a popular ReLe system, while studying the influence that the choice of hypernymy dataset plays in the task.

In summary, we advance towards better understanding ReLe systems by pro-

viding explanations of their predictive behavior and investigating abilities learned by these systems.

Impact Statement

Our research has a direct impact not only on the fields of machine learning (ML) and natural language processing (NLP), but also on the social contexts where such systems may be used. In this thesis, we propose both experimental frameworks where there is an appropriate control of confounding factors and a proper evaluation of the systems' behavior with statistical guarantees of the results obtained. These two contributions may help to pinpoint when the systems have captured possible biases from data and thus take an action in order to avoid the systems exploiting these biases, such as gender or ethnicity biases, which may have serious connotations in society.

Another of our contributions may also help to mitigate the problem mentioned above by designing better datasets; even though we provide analysis of datasets for a particular task, we show what are basic elements to consider when designing a dataset, such as having a background theory of the phenomena under study, or controlling for certain factors. Thus, these basic elements may help to design better datasets that avoid biases such as those mentioned above.

Thus, we hope that the ML and NLP communities will look at this thesis in order to use our methods to a) design better experimental frameworks, b) better evaluate and understand machine learning systems' behavior, and c) design better datasets.

Furthermore, our research may contribute towards linking the fields of machine learning and natural language processing with other scientific disciplines, such as cognitive science, behavioral science, or neuroscience. The fields of ML and NLP can benefit from the research questions, methods, and motivation of these disci-

plines since they share similar objectives, namely to explain certain phenomena occurring in a subject, natural or artificial, and thus to better understand it.

Acknowledgements

I am very grateful to both CONACYT and UCL for the scholarship and studentship respectively provided to me. I thank my supervisor, Sebastian Riedel, for sharing his knowledge and time with me, for guiding me through out all the PhD, and for his patience. I also thank Anthony Hunter for the meetings we had that helped me to see what is the goal of a PhD. I thank so much John Dowell and Anna Korhonen for reviewing this thesis, being my examiners, and providing such a great feedback. I am very happy to have been part of the Machine Reading Group and to have shared good moments with all their members. I'm happy to have shared my PhD journey with my PhD mates Marzieh Saeidi, Matko Bosnjak, Tim Rocktaschel, George Spithourakis, Johannes Welbl, Juan Echeverria, Manal Adham, Manisha Verma, Hugo Lopez, and Marios Constantinides. I thank a lot to Jason Naradowsky, Andreas Vlachos, Guillaume Bouchard, Ivan Vladimir Meza Ruiz, Jeff Mitchell, Pontus Stenetorp, and Pasquale Minervini for all the questions they answered to me, the great discussions we had, and the time they spent to do all of this. Special thanks to Jeff, Pontus (thanks for proposing the term ReLe), and Pasquale for providing feedback of this thesis. I thank my family with whom I'm in debt for giving me all their support during all these years. Finally, I thank Nan Jiang for all her love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Studying a target subject is the intellectual and scientific task of both posing questions about phenomena concerning to the subject itself and to seeking answers for such questions. Common questions across scientific disciplines, about a subject, concern the behavior manifested under specific circumstances and the knowledge it may have learned: What are the reasons for the subject's behavior? Is the subject affected by any external factor in the environment? What are possible internal factors influencing its behavior? What is a cognitive mechanism involved in the observed behavior? Has the subject learned any ability? Different disciplines seek to experimentally answer one or more of these questions. Each discipline targets different levels of abstraction of the same phenomena. Cognitive science works at the representation and functional levels;[1] behavioral science works at the stimulus-response level; neuroscience works at the neural level. For example, cognitive science seeks to build a functional model of how long-term memory works based on behavior observed from people in laboratory tasks (Barrett, 2014), and also aims to propose plausible algorithms of how vision works (Marr, 2010); thus, some questions it aims to answer are: What is a cognitive model that explains how the information is processed by the long-term memory? What is a plausible algorithm that explains behavioral data of human vision? As another example, behavioral science seeks to explain why people buy more items (and spend more) when using a credit card instead of paying by cheques (Soman, 2001); thus, it seeks to answer the question:

[1] In Section 3.1.3, we propose a close relationship between these two types of analysis in the psychology and cognitive science communities.

How does the choice of payment instrument influence the spending behavior of people? As a final example, neuroscience seeks to decode information from neural activity in the brain of a person, such as whether a person is thinking in a living or a non-living object (Chan et al., 2011); hence, one target question to be answered is: Is it possible to extract information from the brain activity of people?

Methodologies to answer the above questions vary from discipline to discipline, and the explanations provided as an answer to the questions vary in form. These explanations, ultimately, are the vehicle to either gain an understanding of the target subject[2] –why/how does the subject behave in the observed way?– or gain knowledge about a specific aspect –is the ability X within the knowledge repertoire of the subject?

Thus, answering the questions from the examples above will allow us to better understand the subjects under study. In this way, we will understand how the spending behavior of people is affected by specific factors; how the long-term memory is organized and how it functions, and what is a possible model explaining how vision mechanisms process information; to what extent we can decode the neural activity from the brain and extract meaningful information. However, the above questions and examples were specific to living subjects. Then, we pose the question: Can we study, in a similar way, machine learning systems?

More concretely, we are interested in a type of machine learning system, namely representation learning systems. Then, is it possible to formulate similar questions for studying representation learning systems? For example, can we provide a representation-level explanation of the behavior of a representation learning system? Can we know what external and internal factors influence the predictive behavior of a system? Can we extract information from the internal components of a system? In order to answer these questions, can we borrow methodology from cognitive science, behavioral science, or neuroscience? Is it possible to draw similar explanations to those from such disciplines in order to better understand these systems? In this work, we will seek answers to these questions.

[2] As it has been argued in the philosophy of science (Grimm, 2010).

In this thesis, we study specific instances of machine learning models dedicated to learning a representation of information. The representation of information deals with ways of encoding information from the real world so that a machine learning system can *understand* this information and operate upon it for a task. A field in machine learning –representation learning– has proposed both new representations and new ways of learning such representations. The new representations proposed are called vector representations, or *embeddings*, which distribute the information learned along several dimensions, i.e. they are representations in the form of continuous vectors. One of the proposed ways to learn the vector representations is via end-to-end machine learning systems; such systems are called *representation learning systems* (ReLe systems). Representation learning systems are widely used in several Natural Language Processing (NLP) tasks for learning the representation of words, sentences, or objects; for example, in question answering (Weissenborn et al., 2017), natural language inference (Chen et al., 2017), and knowledge base population (Riedel et al., 2013).

The use of ReLe systems in these tasks has contributed to achieving state-of-the-art results. The performance of ReLe systems has motivated further development of both representation learning systems and vector representations. However, the complexity of the proposed ReLe systems and the opaqueness of the representations have inhibited the understanding of the behavior of these systems and the phenomena being captured in the representations learned. Thus, actions such as anticipating how a change in a variable of the system affects other variables, knowing what the system has captured from the data, validating that the behavior observed is correct and not an abnormal behavior, or knowing the direction to take in order to improve the system becomes difficult without a proper understanding of the system. Therefore, not fully comprehending and knowing the phenomena occurring in ReLe systems and their learned representations may inhibit scientific progress in the right direction.

Previous work has analyzed both ReLe systems and vector representations in order to understand qualitative aspects of both that are not easy to see at a simple

glance. In particular, it has focused on three main aspects: *Interpretability*: How to explain predictions of ReLe systems in terms of interpretable models? *Robustness*: How robust ReLe systems are to alterations in the input domain? And *abilities learned*: What phenomena, specially linguistic, has been learned by ReLe systems?

Research on interpretability has sought ways of extracting the knowledge encoded in an ReLe system in the form of a proxy model that is interpretable to a human in order to explaining its predictive behavior (Murdoch and Szlam, 2017; Yang et al., 2015; Ribeiro et al., 2016; Lei et al., 2016; Craven and Shavlik, 1995). Analyses on robustness of ReLe systems have evaluated the response of these systems to challenging instances crafted with the objective to measure their generalization ability under difficult circumstances. These analyses have been done for the tasks of reading comprehension (Jia and Liang, 2017), syntactic parsing (B. Hashemi and Hwa, 2016), and machine translation (Isabelle et al., 2017). In addition, previous work has investigated whether ReLe systems are able to capture certain linguistic phenomena, for example hypernymy (Weeds et al., 2014; Roller et al., 2014; Vylomova et al., 2016; Fu et al., 2014; Roller and Erk, 2016), despite the fact that this semantic relation was not explicitly part of the loss function for training the systems.

We claim, however, that there are still open problems to be solved in the task of understanding ReLe systems. One problem is the suitability of previous methodologies for analyzing other types of ReLe systems: It is not clear to what extent we can apply such methodologies to interpret the predictions of, for example, a matrix factorization system (MF), a type of system not considered before in the literature. The main challenges stem from structural and design characteristics of the MF system, which are different to the characteristics of previous ReLe systems studied. Other challenges stem from the scalability and expressiveness of previous interpretable models since these models were used, mainly, for explaining the predictive behavior of simpler ReLe systems. Another open problem is the analysis of the behavior of natural language inference ReLe systems. There is little research on evaluating how robust these systems are to meaningful alterations in the input space, i.e. how

well they can cope with challenging instances. Advancing from this front can help to shed light on what are possible abnormal behaviors, or biases, that a system has captured, and what are the factors that influence the decisions of such systems. A third open problem is the analysis of ReLe systems to encode linguistic phenomena, for example hypernymy. Previous work has provided some evidence for word embeddings capturing this semantic relation; however, it still remains unclear, due to confounding factors in previous experimental frameworks, to what extent it is possible to extract this semantic relation from the representations learned by a ReLe system.

In this thesis, we aim to advance research in *intepretability*, *robustness*, and *abilities learned* by addressing the open problems described above, and thus to better understand the behavior of ReLe systems and the abilities they have learned. We propose to do so via three different types of analysis, namely *representation-level*, *behavioral*, and *internal*. We choose these three types of analysis motivated by work in cognitive science, behavioral science, and neuroscience. In this way, each type of analysis is devoted to one problem under study.

More concretely, we present three case studies; in our first study, we aim to propose a plausible model that explains the input-output process by which a specific matrix factorization system arrives at an observed decision, i.e. we aim to explain the predictions of this system. We do so by learning an interpretable proxy model that faithfully captures the knowledge of the target ReLe system. Furthermore, we show that this proxy model is a type of an explanation at the representation level that describes the decision process of the ReLe system, tying the task of interpretability to that of representation-level analysis from the cognitive science (Marr, 2010).[3] (We note that we are not the first to propose an interpretability analysis to machine learning systems, but we are the first to tie this type of analysis with an analysis from cognitive science.)

In our second study, we evaluate the robustness of three ReLe systems (one

[3]This notion of explanation goes also in accordance to previous work in the philosophy of science and cognitive science. Our proxy model is a type of graphical model which has been considered as an appropriate device to explain how certain phenomena occur (Pacer et al., 2013; Woodward, 2008); in our case, our target phenomenon is the predictive behavior of a matrix factorization system.

of them being a state-of-the-art system) for the task of natural language inference. We analyze to what extent the systems are able to generalize to purposely-crafted instances ranging from easy to those that represent a challenge to the system. To do these analyses, we borrow methodology from the behavioral science; we compute statistical relations between stimulus and response variables and we offer an account of what these relations mean in terms of robustness, i.e. we test whether certain external and internal factors influence the behavior of the systems impacting on their robustness. In this way, we provide explanations[4] of why the systems behave as observed in terms of external and internal factors. Furthermore, we investigate any behavioral patterns in common among the three systems that arise due to the factors under analysis. (We note that, in this case and to the best of our knowledge, we are the first to analyze ReLe systems using methods from the behavioral science.)

In our final case study, we analyze to what extent a specific (and widely used) ReLe system can learn a linguistic relation between two concepts without any explicit signal in its training regime. The level of analysis we are interested in this study is internal, where we open up the system and directly work on specific parts of it, regardless of any stimulus. This study is analogous to work in neuroscience where the study of a phenomenon is carried out via analysis of internal, basic components of the subject under study that give rise to the phenomenon, namely neurons. The outcome of our study is thus not an explanation of how our target system has learned the linguistic relation, but rather we gain knowledge of whether our ReLe system has acquired the linguistic ability suspected. More specifically, we train supervised classifiers that use the parameters of the ReLe system as features in order to figure out if it is possible to find hypernymy as a pattern encoded in these parameters. Even though these classifiers do not work as a proper explanation of how hypernymy is encoded in the ReLe system under study, we do provide an

[4]Statistical associations as a type of explanation in the natural and social sciences has been a central point of debate in the philosophy of science. For example, Turner (2013) claims that such a statistic, along with a description of itself, works as a part of an explanation, but not as a final endpoint for understanding a target phenomenon. In contrast, Strevens (2013) claims that such a setup counts as a complete explanation since an account of a relationship between a proposition (stimulus) and the explanandum (response) is given, it is just the case that the relationship is in a statistical rather than in a deductive form.

explanation of other aspect inherent to our experimental design, namely why some data used to train the classifiers are not useful and why other data are useful. (We note that we are not the first to propose an analysis for extracting hypernymy from the embeddings of a ReLe system, but we are the first to draw a parallel of this type of analysis with analyses from neuroscience.)

Moreover, our findings across our case studies will provide pieces of evidence for *demystifying* certain capacities and abilities of ReLe systems while corroborating others. More concretely, our results show that ReLe systems seem to predict based on correlations and biases found in the data, compromising their generalization ability; and when we test ReLe systems with challenging instances, they significantly lose accuracy; in this way, we show that performance from a test set is not a clear indication of generalization abilities. Also, we find it unclear to what extent ReLe systems encode the capacity to understand language when trained for a task that requires to do so, namely natural language inference, since the systems that we study predict based on factors that have nothing to do with language. Nevertheless, our analyses indicate that ReLe systems seem to be able to encode some type of semantic information, such as hypernymy, based on two different experiments, thus corroborating previous hypothesis from the NLP community.

Finally, we propose to present these three case studies in an isolationist way.[5] As we stated before, studying a target subject can be done at different levels of abstraction. Each level is usually owned by a different discipline, and hence different goals, methods, research questions, and perspectives govern in such a level. Interwining the results derived from each level of analysis is such a complex task that remains as an open problem. Thus, we opt for not integrating the results that we obtain into an unified explanation. Furthermore, we choose to study various ReLe systems at three different levels, namely functional, behavioral, and internal in order to provide a wide understanding of this class of machine learning system instead of, for example, providing either a wide understanding of a single system or a single

[5] An isolationist view of explanations provided by different disciplines is one where each explanation, by itself, adds understanding of certain phenomena and cannot be integrated with other explanations into an unified explanation (Gijsbers, 2016).

perspective of various systems. Hence, this thesis aims to contribute in the scientific understanding of ReLe systems and build more ties with other disciplines.

1.1 Objectives and Research Questions

In this thesis we aim to better understand the behavior of ReLe systems and certain phenomena captured by them; to do so, we aim to investigate both methods for studying the systems and ways of explaining the target behavior. These objectives lead us to global research questions that we will answer through out the thesis:

1. What instruments (research questions, methods, analysis) can we borrow from other disciplines?

2. What type of explanations can we provide to understand the behavior of ReLe systems?

3. In our first case study, what is a good proxy model to represent the decision process of a specific ReLe system?

    - Can we inspect our target ReLe system using methods from previous work?
    - Are interpretable models from the literature suitable for capturing the predictive behavior of the ReLe system?
    - What class of interpretable model may be an equivalent of the ReLe system?
    - Is it possible to view work on interpretability of machine learning systems as a type of a representation-level analysis?

4. In our second case study, how robust are the ReLe systems under analysis to challenging instances and what are possible factors affecting their predictive behavior?

    - What type of analysis can help us discover associations among the response of the systems and the factors?

- Are the systems able to generalize to our challenging instances as well as test accuracy indicates?
- Are there any common behavioral patterns emerging from the systems?
- To what extent work from behavioral science aids in the understanding of the predictive behavior of ReLe systems?

5. In our third study, can we extract linguistic information from a ReLe system?
   - How well can we extract hypernymy relations from the ReLe system under study?
   - How does different hypernymy datasets influence the analysis?
   - What are important characteristics that an useful dataset should fulfill?
   - To what extent work on internal analysis from neuroscience aligns to work on extracting linguistic information from ReLe systems?

1.2 Scope and Limitations

The scope of this thesis is the study and explanation of qualitative aspects of different ReLe systems, such as behavior and abilities learned. We do not aim to improve either the ReLe systems or the representations learned by the systems; however, we believe that our results may help for improving the systems, the representations, and the ways of evaluating their qualities.

A limitation of this work is the restricted number of ReLe systems under study. We study five systems in total, one system in Chapter 4, three systems in Chapter 5, and one system in Chapter 6. Thus, the results that we obtain are specific to those systems and they may not generalize to other ReLe systems. Furthermore, we focus only on three NLP tasks, namely knowledge base population (Chapter 4), natural language inference (Chapter 5), and hypernymy prediction (Chapter 6). Studying these ReLe systems on other NLP tasks may help to better understand them. In addition, as we just said, our ReLe systems are NLP systems; we did not study any ReLe system outside this field, such as computer vision systems, or multi-modal

systems that learn based on both language and visual features. Finally, another limitation is the fact that we present the results from each study in an isolationist way; we do not integrate these results to form a unified explanation of ReLe systems due to the difficulty of such an endeavour.

1.3 Contributions

Our main contributions are analytic since we aim for a better understanding of both the knowledge learned by and the behavior of ReLe systems. Thus, we contribute with new types of explanations; one of them is a graphical model as an equivalent of a representation-level explanation, and the other one is a frame that encompasses statistical correlations along with their descriptions as a type of behavioral explanation. We also contribute with providing a better experimental framework to elucidate whether a linguistic relation is encoded in a system. More specific contributions are the following.

1. First case study:

    - We show that symbolic approaches are not suitable for describing the decision process of a specific matrix factorization system.
    - We provide strong evidence for a Bayesian network being an equivalent of the matrix factorization system under study. Thus, with this Bayesian network we are able to visualize the decision process of the ReLe system and to understand how the system made a prediction.
    - By showing that a Bayesian network is a plausible model to explain behavioral data from the matrix factorization system under study, we provide a piece of evidence for the MF system learning (and possibly predicting based on) correlations among features found in the data.
    - We tie the class of interpretable models with representation-level analysis from cognitive science; i.e., we show that the former analysis can be seen an equivalent of the latter analysis. (We clarify, however, that interpretability analysis is not our contribution; we contribute with tying

the two types of analysis.)

2. Second case study:

   - We provide an account of how the behavior of some ReLe systems is affected by both slight alterations in the input domain and internal/external factors. These analyses provide strong evidence against the picture portrayed, in term of generalization abilities, by over-optimistic test accuracy scores; in other words, our analysis show that the ReLe systems under study are not as robust as previously thought.
   - With the above account, we provide a piece of evidence for ReLe systems classifying instances based on confounding factors that have nothing to do with the task. This evidence may work against the idea of ReLe systems understanding the semantics of the instances. (However, this is only a piece of evidence, and in order to thoroughly and clearly prove any hypothesis about the capabilities of ReLe systems, more studies should be done.)
   - We borrow a methodology and an experimental framework from the behavioral science in order to provide the explanations that show how the systems are influenced by certain factors that act as confounding factors. By borrowing such elements, we provide *internal validity* of our results; i.e., we provide (statistical) confidence that the response of the systems is affected by the factors under study in the way shown by the corresponding explanation. We believe to be the first to borrow methods from the behavioral science to analyze ReLe systems at the stimulus-response level.

3. Third case study:

   - We provide evidence for the ability of a ReLe system to capture hypernymy despite not being explicitly trained to do so. This piece of evidence seems to show that a ReLe system is able to capture a type of

semantic information from a corpus. (However, we abstract away from any claims that ReLe systems are able to capture the semantics of concepts. It is not clear whether our piece of evidence works towards the idea that hypernymy is captured due to the semantics learned by a ReLe system or just due to statistical patterns in the corpus. In order to prove any theory, more studies should be done.)

- We contribute with an analysis of hypernymy datasets crafted for the task of extracting hypernyms. We show that out of six datasets only one is useful for this task given two key characteristics in its design. Furthermore, we show how these two key characteristics are important for building new hypernymy datasets.
- We also show that the selection of scoring function is an important factor in order to interpret the results.
- Even though we borrow the methodology we use to extract hypernymy from the NLP literature, we contribute with drawing a parallel between this type of analysis from NLP with that from neuroscience (brain decoding), and we show how both types of analysis are similar in both objectives and methods.

1.4 Publications

We show the publications derived from these studies:

Chapter 4:

Ivan Sanchez, Tim Rocktaschel, Sebastian Riedel, Sameer Singh. Towards Extracting Faithful and Descriptive Representations of Latent Variable Models. *AAAI Spring Symposium on Knowledge Representation and Reasoning*. March 2015.

Ivan Sanchez Carmona, Sebastian Riedel. Extracting interpretable models from matrix factorization models. *COCO'15 Proceedings of the 2015th International Conference on Cognitive Computation: Integrating Neural and Symbolic Approaches*. December 11 - 12, 2015. Montreal, Canada.

Chapter 5:

V. Ivan Sanchez Carmona, Jeff Mitchell, Sebastian Riedel. Behavior Analysis of NLI Models: Uncovering the Influence of Three Factors on Robustness. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. 2018. Association for Computational Linguistics.

Chapter 6:

V. Ivan Sanchez Carmona, Sebastian Riedel. How Well Can We Predict Hypernyms from Word Embeddings? A Dataset-Centric Analysis. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. 2017. Association for Computational Linguistics.

Chapter 2

Background

In this chapter, we explain the background concepts required to understand the rest of the thesis. We divide this chapter in four sections: We first briefly describe what is representation learning, and then we describe the representation learning models upon which our ReLe systems under study are built on, namely matrix factorization (MF), long short term memory network (LSTM), and bidirectional long short term memory network (Bi-LSTM). We also describe the representations that some NLP systems learn, namely word embeddings; understanding this representation is a requisite before we study another of our target systems. (We describe our systems under study in the corresponding chapters where we describe our research.) In the subsequent sections, we describe some machine learning models, such as decision trees, logistic regression, and support vector machines; as well as probabilistic models, namely Bayesian networks; and symbolic models, such as logic rules upon which we build our research in the next chapters.

We require the understanding of the above mentioned models for the following reasons. In Chapter 4, we use three models that will help us to understand the predictions of a matrix factorization system, namely logic rules, decision trees, and Bayesian networks. Thus, it is relevant to study in this chapter how we induce (learn) an MF model and each of the models that will serve to understand the MF under study; also, we explain how we use these models once learned, i.e., how we make predictions using the trained models. In Chapter 5, we analyze the robustness of ReLe systems built on LSTMs and bidirectional LSTMs; thus, in this chapter

we explain the architectures of both types of models and the set of the equations that characterize them. Finally, in Chapter 6, we extract semantic information from the parameters of a ReLe system; these parameters are commonly known as word embeddings, and they are a way of representing words (concepts). Therefore, in this chapter we explain in detail what are word embeddings, how they are learned, and ways of evaluating their usefulness. Furthermore, in Chapter 6, in order to decode information from word embeddings, we use two classifiers, namely logistic regression and support vector machines, which we also explain in this chapter.

2.1 Representation Learning

The term *representation learning* refers to learning an encoding of data in a form suitable for a machine learning system to operate upon it; such an encoding is the *representation* learned. And the system learning such representation is what we call a *representation learning system* (ReLe) (Bengio et al., 2013).

A common way a ReLe system learns the representation of input data is by minimizing a loss function for either a downstream task, such as knowledge base population or natural language inference, or a language model task. The representations learned are encoded in some of the parameters of the system; these representations may correspond to words, sentences, or other objects depending on the task the ReLe system is trained for.

2.1.1 Representation Learning Models

In this section, we explain three representation learning models widely used in the NLP community, namely matrix factorization, long short term memory network (LSTM), and bidirectional long short term memory (Bi-LSTM). These models have been implemented into several NLP systems; however, understanding their predictions have become a difficult task. In Chapters 4 and 5, we analyze ReLe systems built using the mentioned models, thus it is important to explain the basics of these models to better understand those chapters. More concretely, we explain what are each of these models; i.e., we explain the equations that form these models, what are

they used for, how we can learn them,[1] and how we can use them to predict. Nevertheless, we leave the details of the systems under study to the above chapters; in this section, we describe the characteristics of the models, not the implementations to study.

2.1.1.1 Matrix Factorization

A matrix factorization (MF) is a type of representation learning model whose objective is to populate, or approximate, a matrix of data $\mathbf{X}_{m\times n}$ by learning low-dimensional[2] representations of rows and columns of this matrix.[3] According to (Singh and Gordon, 2008), the mathematical form of an MF model is:

$$\mathbf{X}_{m\times n} \approx f(\mathbf{U}_{m\times k}\mathbf{V}'_{k\times n}) \tag{2.1}$$

We can re-write Equation 2.1 as $\mathbf{Y} = \mathbf{U}_{m\times k}\mathbf{V}'_{k\times n}$, where the matrix $\mathbf{Y}$ is a reconstruction, or approximation, of the matrix $\mathbf{X}$. In $\mathbf{Y}$, all the cells have been populated by means of the factors $\mathbf{U}$ and $\mathbf{V}$.[4] Function f is particular for the task where the MF model is going to be applied. For example, in Chapter 4, our system under study uses a sigmoid function.

The factors $\mathbf{U}$ and $\mathbf{V}$ contain the low-dimensional representations learned for each row and column of the matrix $\mathbf{X}$; i.e., each row vector of both $\mathbf{U}$ and $\mathbf{V}$ corresponds to a low-dimensional embedding. Figure 2.1 shows a graphical depiction of an MF model; in this figure, a row $\mathbf{X}_i$ is represented by the 2-dimensional embedding $\mathbf{U}_i$, and similarly, the representation of a column $\mathbf{X}_j$ is encoded in the 2-dimensional vector $\mathbf{V}_j$.

Learning the matrix $\mathbf{Y}$ can be done by minimizing a loss function that computes

[1] We succinctly explain learning methods, but we do not aim to go to a fine-grained detailed since our main goal is to understand pre-trained systems, as we will study them in Chapters 4 and 5; thus, we prefer to focus more on the characteristics of the models, rather than on learning algorithms.

[2] The term *low-dimensional* is used to denote that the size of the representation learned (in the form of a continuous vector) is lower than the size of a one-hot vector representation of the same object.

[3] In this matrix of data, each row and each column represent an object. For example, suppose $\mathbf{X}$ is a dataset of recommendations for users, then a row $\mathbf{X}_i$ represents a user $user_i$ while a column $\mathbf{X}_j$ represents an item $item_j$.

[4] The notation $\mathbf{V}'$ indicates the transpose of the matrix $\mathbf{V}$.

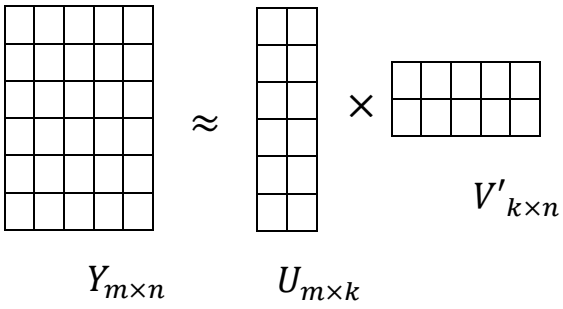


Figure 2.1: A matrix factorization model. Matrix **X** is a matrix of data used as training data for learning the factors **U** and **V**.

the distance between the matrix of data **X** and its approximation:

$$\sum_{i,j}(x_{ij} - y_{ij})^2 \tag{2.2}$$

This loss function is minimized by an optimization algorithm, such as gradient descent. The parameters learned are the factors in which the matrix **Y** is decomposed, namely **U** and **V**.

Inference in an MF model accounts for populating the cells of the matrix **Y** and is done via a dot product between the factors learned, as shown in Equation 2.3, where function f is the same function as in Equation 2.1. We note that inference can only be done for rows and columns of **Y** for which factors **U** and **V** were learned at training time.

$$y_{ij} = f(\langle \mathbf{U}_i, \mathbf{V}'_j \rangle) \tag{2.3}$$

2.1.1.2 Long Short-Term Memory

A Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is another type of representation learning model widely used in several NLP tasks. This model learns the embeddings of both words and sentences in an end-to-end fashion. And it is able to keep track of long-term dependencies in a sentence while avoiding a common problem found in other models, namely the *vanishing gradient* (Good-

fellow et al., 2016). This problem occurs as a side of effect of the learning algorithm –*backpropagation*– that tries to propagate an error signal through the many layers of the model; this signal diminishes its intensity as it passes through the layers up to the point where it becomes negligible. An LSTM solves this problem through the use of *gates* that control the flow of information allowing the error signal to be fully propagated through all the layers. Furthermore, the architecture of an LSTM allows for an end-to-end, joint learning of both the parameters of the gates and the input embeddings using a single loss function.

The architecture of an LSTM is that of a recurrent neural network (RNN), i.e. it receives self-feedback. At each time step $t \in 1,..,T$ the LSTM receives an input embedding $\mathbf{x}_t$, and the previously computed hidden state $\mathbf{h}_{t-1}$ and memory cell $\mathbf{c}_{t-1}$, then it computes the current hidden state $\mathbf{h}_t$ and memory cell $\mathbf{c}_t$ through the set of gates $\mathbf{i}_t$ (input), $\mathbf{o}_t$ (output), and $\mathbf{f}_t$ (forget). The recurrence comes from both the hidden state and memory cell. If a classification output is required, then a *softmax* layer is placed on top of the last hidden state $\mathbf{h}_T$.

$$\mathbf{i_t} = sigmoid(\mathbf{w}_{xi}\mathbf{x}_t + \mathbf{w}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.4}$$

$$\mathbf{o_t} = sigmoid(\mathbf{w}_{xo}\mathbf{x}_t + \mathbf{w}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.5}$$

$$\mathbf{f_t} = sigmoid(\mathbf{w}_{xf}\mathbf{x}_t + \mathbf{w}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.6}$$

$$\mathbf{g_t} = tanh(\mathbf{w}_{xg}\mathbf{x}_t + \mathbf{w}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g) \tag{2.7}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{2.8}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t) \tag{2.9}$$

The set of Equations 2.4-2.9 form the structure of a cell, a complex structure by itself. The memory cell (Equation 2.8) acts as a memory by receiving information from the past memory cell ($\mathbf{c}_{t-1}$). This memory is regulated (controlled) by the forget gate which decides to what extent the past information is going to be stored. The forget gate is a feed-forward neural network that based on the input vector $\mathbf{x}_t$ and the past hidden state $\mathbf{h}_{t-1}$ learns the degree, $\mathbf{f}_t \in [0,1]^n$, to which the past cell

state information should be forgotten or remembered. In addition, the information coming from the past memory cell is combined with the information coming from the neural network $\mathbf{g}_t$ controlled by the input gate $\mathbf{i}_t$ which regulates the extent to which the information is read. The output gate $\mathbf{o}_t$ controls the flow of information that is going to be passed to the next cell in the LSTM. This gate is also a feed-forward network, similar to the forget and input gates.

It is important to note that in a cell each gate has its own set of parameters $`_x$ (input weights), $`_h$ (recurrent weights), and $\mathbf{b}$ (biases), all of them learned by backpropagation. And these parameters are shared through time; i.e., the cell $\mathbf{c}_t$ uses the same parameters as the cell $\mathbf{c}_{t-1}$. Another set of parameters also learned are the input vectors $\mathbf{X}$. In an NLP task, one input vector $\mathbf{x}^k$ is usually a sentence, and it is formed by a set of words, $\mathbf{x}^k_1, ..., \mathbf{x}^k_T$, each of which is represented by a word embedding, i.e. a vector $\mathbf{x}^k_t \in \mathbb{R}^n$. Altogether, the number of parameters is usually around the hundred of thousand, or even in the millions, depending on the task and the size of the dataset.

2.1.1.3 Bidirectional LSTM

A bidirectional LSTM (Bi-LSTM) is simply the union of two LSTMs in order to capture more information than a single LSTM can do. Each LSTM in a Bi-LSTM, $\mathbf{LSTM}_1$ and $\mathbf{LSTM}_2$, reads the input in a different way. One LSTM reads the input forwards and the other reads the same input backwards (Goodfellow et al., 2016). For example, an $\mathbf{LSTM}_1$ will encode the sentence *The cat eats* into a sentence embedding by reading the words in the following order: *The*, *cat*, *eats*, whereas an $\mathbf{LSTM}_2$ will process the same sentence following the opposite order, *eats*, *cat*, *The*. The embeddings learned by the two LSTMs summarizing the whole input sequence are then used in different contexts, such as transfer learning or classification. If the target task is classification then a *softmax* layer is fed with the outputs from the two LSTMs and it outputs class-label probabilities.

Learning the parameters of the two LSTMs is done jointly via an optimization algorithm, such as gradient descent. For a classification task the most common loss function to minimize is the cross-entropy loss where the output from the soft-max

is compared against the gold class label.

2.1.2 Representations of Words

In this section, we describe word embeddings, which are representations of words learned by some ReLe systems. We explain some methods of how these vector representations can be learned and evaluated. We also, explain how they encode some semantic information, such as similarity. Word embeddings are highly used in recent NLP systems, due to their usefulness in encoding similarity which allows NLP systems to improve accuracy on tasks such as question answering or natural language inference. For example, knowing that the concept of *dog* is similar to the concept of *animal* may help a system to do an entailment. However, it is not clear if this type of representation can encode more information than just similarity, a research question which we will answer in Chapter 6. Thus, in this section, we review the fundamentals of word embeddings required to understand the analysis of a particular set of word embeddings obtained from a pre-trained ReLe system in Chapter 6.

2.1.2.1 Word Embeddings

Word embeddings, also known as low-dimensional vectors or distributed representations, are dense vectors that represent the *meaning* of words; i.e., each word w_i in a vocabulary V is associated with a continuous vector $\mathbf{w}_i$ of dimensionality $d = k$ (Goodfellow et al., 2016). For example, the word *cat* is represented by a word embedding in $\mathbb{R}^k$, such as $\mathbf{w}_{cat}$.

Representing words using embeddings has an advantage with respect to using symbolic representations where a word is associated with a one-hot vector (Goodfellow et al., 2016):[5] *semantically* similar words are close in the embedding space in comparison to *semantically* different words. For example, the embedding of *cat* is closer[6] to the embedding of *dog* than to the embeddings of unrelated concepts,

[5] An example of a one-hot vector for the word *cat* is the vector $\mathbf{v}_{cat}$: [0 0 ... 0 1 0 ... 0], where the size of $\mathbf{v}_{cat}$ is the number of elements in the vocabulary V; i.e., each word in the vocabulary is represented by a position in the vector representation.

[6] The notion of distance between embeddings can be quantified using a metric, being the cosine and Euclidean distances two of the most popular.

such as *moon* or *train*. The rationale behind this statement is based on the *distributional hypothesis* (Harris, 1954; Firth, 1957). The main idea of this hypothesis states that if two words in a text occur along with same contextual words, then they have similar meaning. For example, *cat* and *dog* may occur with the words *pet*, *veterinary*, or *animal*, all of them characteristic of the concept *domestic animal*; then, the vectors of *cat* and *dog* are more similar to each other than to vectors of other concepts outside the category *domestic animal*.[7] One question that may arise at to this point is how do we get similar concepts to be close in vector space?

Each dimension of a word embedding corresponds to one parameter of a representation learning system. As we saw in Sections 2.1.1.1 and 2.1.1.2, we learn embeddings by minimizing a loss function for solving a task. There are two types of tasks in which we can learn word embeddings, downstream and auxiliary.

Learning word embeddings via a downstream task, such as machine translation, may require more resources than learning them via an auxiliary task. The main bottlenecks reside in the corpus and the time required to train. Usually, the final use of word embeddings are as features in downstream tasks; in this sense, it is reasonable to simply use such tasks to learn the embeddings. However, such pre-trained embeddings, specific to a task, may not be of use as *off-the-shelf* features for another different task, due to the domain information that they captured from the downstream corpus. For example, embeddings learned using a machine translation objective may capture linguistic information probably not required for the task of question answering, and information that is necessary for QA may not be present in the MT corpus. Of course, one may train embeddings specific for QA using the corpus of the task of interest. However, there is evidence for the usefulness of using pre-trained word embeddings in downstream tasks: when the corpus of the target task is small, embeddings can *transfer* information learned from the task they were optimized for; furthermore, they help to reach local minima not reachable by a random initialization of the parameters.

[7] We may speculate that the characteristics shared by two concepts in a text are captured by the word embeddings along the k dimensions. Previous work has successfully extracted some information from pre-trained embeddings, but it is still an open question to what extent embeddings can capture semantic, syntactic or other type of information.

A common auxiliary task for learning word embeddings is a variant of language modelling (Goldberg, 2016). The task is either to predict a focus word w_i given a context c_i,[8] $p(w_i|c_i)$, or a context given a focus word, $p(c_i|w_i)$. Advantages of this auxiliary task over downstream tasks are a) the abundance of text corpus, since documents in plain language, such as Wikipedia articles, serve as a training corpus; b) the task is unsupervised in the sense that no labels (and therefore no manual annotations) are required; c) usually less training time is required; d) the word embeddings are not specific to a downstream task, but rather to the domain of the corpus, and the bigger the corpus the more information it can capture.[9] In addition, some works in NLP have found useful using word embeddings obtained from auxiliary tasks (Weissenborn et al., 2017; Chen and Manning, 2014; Parikh et al., 2016).

Evaluating word embeddings can be done in two ways: in a downstream task (extrinsic evaluation) or in a proxy sub-task (intrinsic evaluation.) A proper evaluation of embeddings in a downstream task requires an adequate control of the components of the NLP system that may have an effect on the global accuracy; therefore, this type of evaluation has a certain degree of difficulty depending on the complexity of the NLP system. On the other hand, in an intrinsic evaluation, we can evaluate, to some extent, how well the embeddings have captured desired aspects; this type of evaluation is performed in a sort of *laboratory* conditions and is usually easier to control for, given a properly crafted dataset.

Two common intrinsic tasks are *similarity* and *analogy*. In the *similarity* task, it is measured if the similarity of two concepts, according to human judgment, is captured in the corresponding word embeddings. For example, in the *wordsim353* dataset[10] *tiger* and *cat* are similar by 7.35 points (in the scale of 0 to 10), while *plane* and *car* are so by 5.77 points; therefore, if the distance of $\mathbf{w}_{tiger}$ to $\mathbf{w}_{cat}$, in vector space, is closer than that of $\mathbf{w}_{plane}$ to $\mathbf{w}_{car}$, then we assert that the embeddings have captured some notion of similarity. On the other hand, the *analogy*

[8] A set of surrounding words to the left and right of the focus word if the context is symmetric.

[9] However, if the corpus is specific to a domain, such as news, then it may not contain a broad class of linguistic phenomena, but only those pertinent to the domain.

[10] http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/wordsim353.html

task of Mikolov et al. (2013) measures, to some extent, if the embeddings have captured a sense of hypernymy: *king* is to *man* as *x* is to *woman*; it has been claimed that if the embedding of $\mathbf{w}_{queen}$ can be recovered via the vector operation $\mathbf{w}_{king} - \mathbf{w}_{man} + \mathbf{w}_{woman} = \mathbf{w}_{queen}$ then the embedding space learned has captured a sense of hypernymy.

2.2 Machine Learning Models

In this section, we explain three machine learning models that we use as classifiers, namely logistic regression, support vector machines, and decision trees.

Two of these models will help us to decode information from a set of word embeddings in Chapter 6, namely logistic regression and support vector machines. Here, we explain the fundamentals required to understand how we will use them in the mentioned chapter; i.e., we explain how these models classify an instance as pertaining to one out of two classes. More concretely, we will use these two classifiers to predict if two word embeddings fall in the relation of hypernymy or not; in this way, the classifiers will work as decoders and will allow us to extract hypernymy information from the embeddings.

On other hand, we will use decision trees (DT) in Chapter 4 in order to understand the predictions of a ReLe system; i.e., we will learn decision trees that will mimic the predictive behavior of the target ReLe system. Due to the easiness in understanding the predictions from a decision tree, we use this model as a proxy to understand a more complex system. Thus, we consider important to understand the characteristics of decision trees, how we can induce a DT, and how we can use it to make predictions.

We decided to use these three machine learning models due to their popularity in the literature. Logistic regression and support vector machines have been widely used in the NLP literature to extract different types of semantic information from word embeddings, such as hypernymy and meronymy. Moreover, they have been used to a great extent in the neuroscience also to extract semantic information from readings of neural activity of people. On the other hand, decision trees are a classic

model in the literature of interpretability of complex systems.

2.2.1 Logistic Regression

Logistic regression is a binary, linear classifier which given an input vector $\mathbf{x} \in \mathbb{R}^d$ outputs a number $f(\mathbf{x}) \in [0, 1]$ interpreted as the probability of the input pertaining to a certain class, i.e. $f(\mathbf{x}) = p(y = 1|\mathbf{x})$, where y indicates the class value expected (Murphy, 2012; Mohri et al., 2012; Shalev-Shwartz and Ben-David, 2014). Since logistic regression is a binary classifier, there are only two possible classes, $y = 0$ and $y = 1$. The mathematical form of this classifier is shown in Equation 2.10, where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the parameters to be learned.

$$f(\mathbf{x}) = sig(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + exp(-(\mathbf{w}^T\mathbf{x} + b))} \tag{2.10}$$

Learning the parameters of a logistic regression classifier requires a labelled dataset D, where each instance is of the form $(\mathbf{x}, y)$, and a cross-entropy loss function. This loss function, shown in Equation 2.11, can be seen as a maximum likelihood estimator of the parameters, and can be minimized by an optimization algorithm, such as gradient descent.

$$L(\mathbf{w}, b) = \sum_{i=1}^{N} (y_i) log(f(\mathbf{x}_i)) + (1 - y_i) log(1 - f(\mathbf{x}_i)) \tag{2.11}$$

Inference by this type of classifier is done in a straightforward way by applying the function 2.10 on an input vector $\mathbf{x}_i$ and obtaining a probability estimate $p(y = 1|\mathbf{x})$. This probability value is thresholded at threshold $\alpha = 0.5$ in order to obtain a class value, though different threshold values can be used.

2.2.2 Support Vector Machines

SVMs are another type of binary, linear classifiers. This type of classifiers seeks to maximize the distance between training instances and a hyperplane separating these instances by class label. A key addition to this classifiers is that of a kernel function, K, which allows to learn a non-linear decision boundary in the input space (Mohri et al., 2012; Shalev-Shwartz and Ben-David, 2014). This function provides

the dot product of two instances in a higher-dimensional feature space $\mathscr{F}$,[11] as shown in Equation 2.12 where the function ϕ, intrinsically computed by K, maps from the input space to the feature space. An example of a common kernel function is the Gaussian kernel (Equation 2.13). Computing this dot product in a higher dimensional space facilitates learning a linear hyperplane to separate the instances by class label. Thus, the resulting decision boundary learned is linear in the feature space $\mathscr{F}$ and non-linear in the input space $\mathscr{X}$.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i)), \phi(\mathbf{x}_j)) \rangle \tag{2.12}$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j)) \tag{2.13}$$

Learning the decision boundary in the feature space $\mathscr{F}$ accounts for minimizing the following quadratic program:

$$L(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m y_n y_m K(\mathbf{x}_n, \mathbf{x}_m) \tag{2.14}$$

Subject to the conditions:

$$0 \geq a_n \leq C \tag{2.15}$$

$$\sum_{n=1}^{N} a_n y_n = 0 \tag{2.16}$$

Where each a_n is a Lagrange multiplier, and each instance is of the form $(\mathbf{x}, y)$ where y is a label.

Predicting the label $f(\mathbf{x}_i)$ of a given test instance $\mathbf{x}_i$ is done using Equation 2.17; however, this output is not a probability. In order to estimate such a probability, a logistic regression is fitted using a subset of instances labelled with predictions of the SVM learned (Platt, 1999).

[11] This feature space can even be infinite dimensional.

$$f(\mathbf{x}_i) = \sum_{n=1}^{N} a_n y_n K(\mathbf{x}_i, \mathbf{x}_n) \tag{2.17}$$

2.2.3 Decision Trees

Decision trees (DT) are hierarchical classification models, where given an instance $\mathbf{x}_i$ (a vector of features), a class label y_i is output after analyzing each of the features x_j. A decision tree consists of two basic elements: *Internal* (decision) nodes and *leaf* (classification) nodes. An internal node represents a test on the value of a specific feature of an instance. If a feature x_j, from the input domain, has a Boolean domain then its corresponding decision node, namely d_j, will test, for any instance $\mathbf{x}_i$ to be classified, the value x_{ij} and depending on this value (0 or 1) will be the next decision node to try. If feature $x_j \in \{1,...,m\}$ is a categorical variable then the number of splits in d_j can be either m (one split for each possible value), or 2 (a binary split): $x_{ij} = category_k$, $x_{ij} = any\_other\_category$. In the case of continuous features, a binary split is applied: $x_{ij} < t_j, x_{ij} \geq t_j$, where t_j is a learned threshold (Murphy, 2012; Shalev-Shwartz and Ben-David, 2014).

2.2.3.1 Classifying an instance

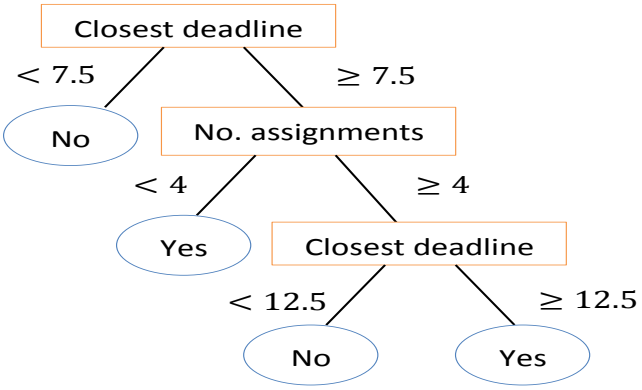
A leaf node classifies a particular instance: Any instance $\mathbf{x}_i$ that falls into a leaf node l_k will be assigned a class label $y_i = k$. So, classifying a particular instance accounts for an ordered test of the values of the features for such instance; in this way, a decision tree is a hierarchical model given that there is a hierarchy in the order of the nodes, starting from the *root* node (first node of the tree) and ending in a *leaf* node. See Figure 2.2 for an example of a decision tree.

2.2.3.2 Inducing a tree

Inducing a decision tree from training data accounts for building the hierarchical structure of the nodes. In other words, learning a decision tree is the recursive function of structuring the features from the input space in a hierarchical tree structure. Since each node in a decision tree makes a split on the training data, we can say that learning a decision tree accounts for learning hyperplanes in the input space that separates instances from different classes, where the first node in the hierarchy

No. Assignments	Closest deadline (days)	Weather	Go to party?
2	9	Sunny	Yes
5	7	Sunny	No
6	15	Rainy	Yes
1	1	Rainy	No
5	10	Sunny	No
3	8	Rainy	Yes

(a) Table with training instances for the concept *Go to party*.



(b) A decision tree for the training data above. Internal nodes test for the value of features *Closest deadline* and *No.* Assignments. Leaf nodes label an instance as class *Yes* or *No* for the concept *Got to party*.

Figure 2.2: A decision tree representation for the concept *Go to party*. The feature *Closest deadline* splits the input space at different levels, given its importance. All the splits are binary. The feature *Weather* does not contribute with an information gain (except in the split ≥ 12.5 where it could have been substituted the node *Closest deadline* for the node *Weather*).

imposes the first margin on the input space. See Figure 2.3 to see an example of the margins induced for the example in Figure 2.2.

An algorithm for learning a decision tree is shown in Algorithm 1. Given that inducing a decision tree from data is an NP-complete problem, the learning algorithms are usually greedy. The algorithm shown here selects in a greedy way the nodes to add to the hierarchy. This selection method usually relies on two measures, information gain and squared error. C4.5 and CART (Murphy, 2012), two of the most popular learning algorithms for decision trees, use these node-selection measures while inducing the hierarchical structure in a similar fashion as the algo-

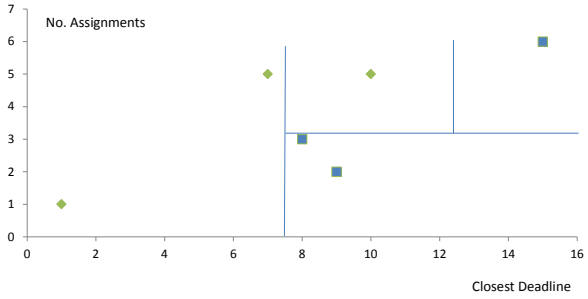


Figure 2.3: Input space divided by the decision tree in Figure 2.2. Blue squares represent class *Yes*, green diamonds represent class *No*.

rithm shown here.

x_j: feature j
initialize: $x_j \leftarrow Null$, $D \leftarrow$ *training dataset*
learnDecisionTree(*portion of data* D_i) {
if *Entropy of output variable* y *in* $D_i == 0$ **then**
  if $y == 0\ \forall(\mathbf{x}, y) \in D_i$ **then**
    return leafNode($y \leftarrow False$)
  else
    return leafNode($y \leftarrow True$)
  end
else
  choose best feature x_j *and create its node*
  $D_{left} \leftarrow (\mathbf{x}_i, y_i) \forall i\, s.t.\, x_{ij} = 0$
  learnDecisionTree(D_{left})
  $D_{right} \leftarrow (\mathbf{x}_i, y_i) \forall i\, s.t.\, x_{ij} = 1$
  learnDecisionTree(D_{right})
end

Algorithm 1: Algorithm for learning a binary decision tree. Each internal node has a binary split, and the class label in a leaf node is also binary (*True, False*). The entropy in a split measures the *homogeneity* of the data, if the entropy is 1 (maximal entropy) it means we have 50% of chance of classifying correctly an instance, if entropy is 0 then all the data instances falling in the leaf are from the same class.

Information gain (IG) is probably the most popular score for node selection. It measures the mutual information between a candidate feature x_j and the class variable y:

$$IG(y,x_j) = H(y) - H(y|x_j) = -E[log_2 p(y)] + E[log_2 p(y|x_j)] = \\ -\sum_k p(y=k) log_2 p(y=k) + \sum_m p(x_j = m) H(y|x_j = m) \tag{2.18}$$

In equation 2.18, H is the entropy of a variable.

Mutual information is a measure of the strength of statistical dependence between two variables in $[0,1]$; the higher the mutual information, the more dependent the two variables are. In terms of data, we can interpret information gain as a measure of the *homogeneity* of the data after feature x_j induces a margin in the input space. For example, consider Figure 2.4 where training data is split after adding the root node from example in Figure 2.2. By adding the feature *Closest deadline* we get the following information gain:

$$IG(y, Closest\,deadline) = H(y) - H(y|Closest\,deadline) \tag{2.19}$$

Where

$$H(y) = -(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1 \tag{2.20}$$

$$H(y|Closest\,deadline) = -(\frac{2}{6}(0 log_2 0 + 1 log_2 1)) - (\frac{4}{6}(\frac{1}{4} log_2 \frac{1}{4} + \frac{3}{4} log_2 \frac{3}{4})) = 0.5408 \tag{2.21}$$

Thus

$$IG(y, Closest\,deadline) = 0.4592 \tag{2.22}$$

This means that by splitting on the feature *Closest deadline*, the uncertainty in the classification of the data is reduced by 0.4594 bits. As we see from Figure2.4, on the left split we are able to correctly classify class *No*, given that the entropy (uncertainty) is zero. On the right branch we are able to classify class *Yes* 3 out of 4

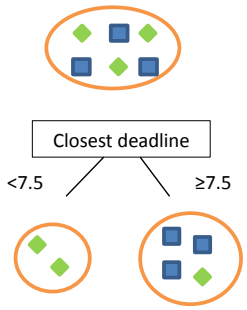


Figure 2.4: Training data division before and after first split on feature *Closest deadline* from example in Figure 2.2. Green diamonds represent class *No*, blue squares represent class *Yes*. Note that the entropy on the left branch is zero, due to the homogeneity of the data.

times correctly.

2.2.3.3 Probability Estimation

An important characteristic of a decision tree is its capability of estimating class probabilities. Estimating the probability that an input instance $\mathbf{x}_i$ pertains to a class $y = k$, $p(y = k|\mathbf{x}_i)$, is done at any leaf node l_m by means of maximum likelihood estimation:

$$p(y = k|\mathbf{x}_i) = \frac{number\_of\_instances\, \mathbf{x}_j \in l_m \, s.t. \, class(\mathbf{x}_j) = k}{Total\_number\_of\_instances \in l_m} \tag{2.23}$$

2.3 Probabilistic Graphical Models

In this section we describe a model that will help us to explain predictions of a ReLe system in Chapter 4. In a similar way as we will use decision trees, from Section 2.2.3, we will use a probabilistic model to mimic the predictive behavior of the target ReLe system; more concretely, we will use a Bayesian network. To the best of our knowledge, we are the first to use this model in order to explain predictions of a ReLe system. Thus, we devote this section to describe and characterize this model, explain how we can learn it, and how we can make predictions.

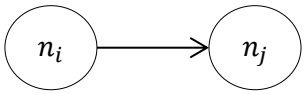


Figure 2.5: A simple Bayesian network consisting of only one parent node (n_i) and one child node (n_j).

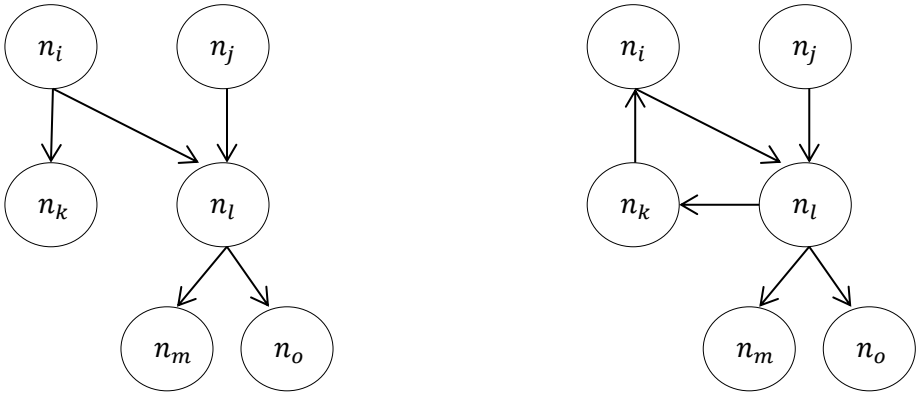
2.3.1 Bayesian Networks

A Bayesian network (BN) is a graphical representation of a factorized joint probability distribution over a set of random variables $x_1, ..., x_n$. Each factor in a BN is called a *family* and it encodes a local conditional probability distribution of a *child* node given its *parent* nodes. Thus a node n_i is a graphical representation of a random variable x_i, and it has a direct influence over a node n_j if there is a direct link from the former node the latter node (see Figure 2.5). This influence is associated to a conditional probability distribution: $p(x_j|x_i)$ (a factor); this conditional probability, graphically represented in Figure 2.5, can be interpreted as node n_i being the parent of node n_j, and consequently n_j being a child node. In a general Bayesian network there is no restriction for a node to have more than one parent, though a node it is not allowed to be its own parent (Koller and Friedman, 2009; Murphy, 2012).

2.3.1.1 General Bayesian networks

Every Bayesian network is characterized by a) a graphical structure (the set of nodes and their links) and b) a set of parameters (the set of local conditional probability distributions). Two restrictions imposed in the structure of a Bayesian network are *acyclicity* and *directionality*: The structure of a BN cannot have cycles and each link between nodes must have a direction; for these reasons, the structure of a BN is called a *DAG* (Directed Acyclic Graph). In Figure 2.6 we can see both an example of a DAG and an example of a graph with a cycle.

An important probabilistic assumption that a BN encodes is *conditional independence*. Two random variables, x_i and x_j, are conditionally independent given



(a) A Directed Acyclic Graph.

(b) A graph with a cycle made up by nodes n_i, n_k, n_l.

Figure 2.6: Examples of a DAG and not a DAG.

a third random variable x_k, denoted by $x_i \perp x_j | x_k$, if and only if $p(x_i, x_j | x_k) = p(x_i|x_k)p(x_j|x_k)$. From a graphical perspective, this is represented by the absence of a link between the nodes n_i and n_j, where both nodes have a connection to, or from, the node n_k. This configuration allows an indirect influence between n_i and n_j through node n_k. In both the *causal*[12] and *evidential* forms of influence, the node n_k *blocks* the influence between n_i and n_j if it is observed, i.e. if the random variable x_k is realized to a value A. In this way, nodes n_i and n_j have no influence over each other (see Figures 2.7a and 2.7c).

A broader concept of conditional independence is *d-separation*, where the nodes n_i and n_j are separated in the DAG by a set of nodes **d** between them. And the influence of one node on the other is blocked if at least one node in **d** is observed. Basic configurations of d-separation are shown in Figure 2.7. We can identify four basic types of d-separation. In a *causal influence* graph the flow of influence starts from the node with the highest hierarchy in a sub-graph and descends through a single path until it reaches a leaf node. We can say that the former node *causally* influences all nodes in the path to the leaf node. The flow of influence is broken at a

[12]The term *causal* refers to the direction of flow of influence between nodes, from parent to child, rather than to a *causal Bayesian network* structure which is created with the purpose of taking into account possible external interventions.

node n_k if this node is observed, and thus all descendents of n_k are no longer under the influence of the root node of the sub-graph.

A similar setting to the causal flow is found in an *evidential influence* graph. In this scenario the flow of influence starts in a child node and propagates upwards to its ancestors in a single path until it reaches the highest node in the sub-graph. Another type of d-separation is called *common cause*; in this configuration, a node n_k accounts for the cause of two or more nodes by acting as a parent node for all of them. A restriction is the lack of links among child nodes. If node n_k is observed then the influence among its child nodes is blocked, making these nodes conditionally independent of each other given their parent node.

The last type of separation is the so-called *v-structure* or *explaining away*. In this structure a node n_k has at least two parents. If node n_k is observed then it allows a *flow* of influence among its parents; thus each of the parent nodes becomes dependent of each other given that each of these is a possible explanation for the observation of node n_k. In this way, each parent *explains away* the rest of the possible causes.

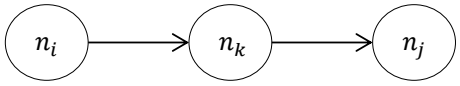
We note that d-separation in a DAG does not necessarily imply conditional independence, as in a *v-structure*. We also note that if no node is observe in DAG, it might be possible that two nodes n_i and n_j can influence each other.

2.3.1.2 Tree-structured Bayesian networks

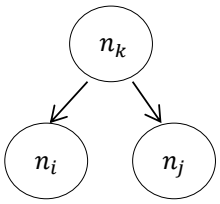
A special case of a Bayesian network is a tree-structured BN where the DAG is in the form of a tree. An example of a BN tree is provided in Figure 2.8. The restriction in the structure of the graph is that any node can have at most one parent, i.e., $|parents(n_i)| \leq 1$, where size zero is left exclusively to the root node. Another characteristic of this type of BN is that no node remains without a link. On the other hand, the number of descendents of a node is unrestricted. In the case that the number of descendents is fixed to be at most 2 then we say it is a binary tree.

2.3.1.3 Learning Bayesian networks

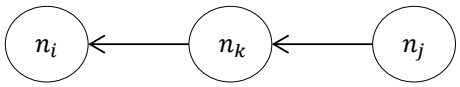
Learning a Bayesian network from data implies both inducing the structure (DAG) and learning its parameters (local conditional probabilities) (Koller and Friedman,



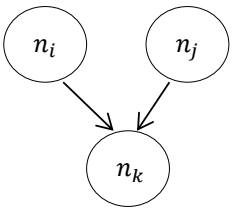
(a) *Causal* indirect influence between nodes n_i and n_j through node n_k. The flow of influence goes from node n_i to node n_j passing through n_k.



(b) *Common cause* indirect influence between nodes n_i and n_j through node n_k.



(c) *Evidential* indirect influence between nodes n_i and n_j through node n_k. The flow of influence goes from node n_i to node n_j passing through n_k.



(d) *V-structure* indirect influence between nodes n_i and n_j through node n_k (also known as *explaining-away*).

Figure 2.7: Basic configurations for conditional independence. In Figures 2.7a, 2.7c, and 2.7b, if noden_k is observed then n_i is separated from node n_j, i.e., the influence is blocked. Opposite, in Figure 2.7d if node n_k is observed then nodes n_i and n_j can influence each other.

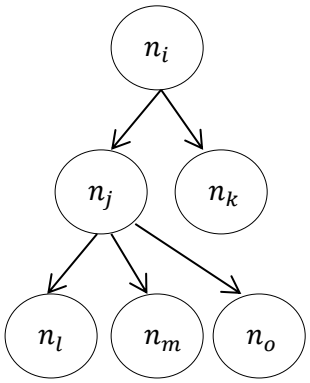


Figure 2.8: Example of a Bayesian network tree. The number of parents of a node is restricted to be at most 1.

2009). A training instance for this task is on the form $\mathbf{x} = (x_1, x_2, ...x_n)$, where $\mathbf{x}$ denotes a vector of independent features (random variables). Thus learning a Bayesian network accounts for inducing a factorized joint probability function where each factor corresponds to a local conditional probability distribution.

Two of the most common approaches for learning the structure are *conditional independence tests* and *structure search*. The first approach usually initializes a fully connected graph and then removes links between nodes that are proved to be statistically independent. A problem with this approach is the difficult to faithfully detect independencies if the size of the sample data is small due to possible random correlations between the variables.

In the second approach, a search in the space of the DAGs is performed by a greedy method, such as hill-climbing, using an scoring function based on information theory, such as mutual information (MI), Bayesian information criterion (BIC), or Akaike Information Criterion (AIC). Maximizing a mutual information criterion has been proved to be equivalent to maximizing the likelihood of the data given a candidate model; however, overfitting arises due to a positive correlation between the number of edges in the candidate model and the mutual information score. On the other hand, BIC and AIC are two model selection measures that alleviate the overfitting problem by penalizing complex structures. In either case, scoring a Bayesian network is done through the *decomposability* property: The total score of a BN can be decomposed as the sum of the local scores for each family, as seen in Equation 2.24:

$$score(DAG_k) = \sum_{i \in N} score(x_i | parents(x_i)) \tag{2.24}$$

Where N is the set of nodes in the candidate DAG and $score(x_i|parents(x_i))$ is the mutual information between a node and its parents.

Once a structure has been selected, learning the parameters accounts for (independently) estimating a local conditional probability function for each family by means of either maximum likelihood estimation (MLE):

$$p(x_i|parents(x_i)) = \frac{\#(x_i, parents(x_i))}{\#(parents(x_i))} \tag{2.25}$$

Where we define the operator # as the number of times its argument its seen in a dataset D:

$$\#[x] = \sum \mathbf{1}\{x \in D\} \tag{2.26}$$

Therefore, the term $\#(X_i, parents(X_i))$ in Equation 2.25 indicates the number of times the random variable x_i is seen along with its parents in the dataset D; similarly, the term $\#(parents(X_i))$ indicates the counts of the parent nodes of x_i in the dataset.

Another approach to learn the parameters is by means of maximum a posteriori (MAP), where suitable priors are either a Laplacian smoothing or a Dirichlet function:

$$p(x_i|parents(x_i)) = \frac{\#(x_i, parents(x_i)) + \alpha}{\#(parents(x_i)) + \alpha(|domain(x_i)|)} \tag{2.27}$$

Where α is a constant indicating an arbitrary number of pseudo-counts.

2.3.1.4 Inference in Bayesian networks

Variable elimination is an algorithm for exact inference of the type $p(\mathbf{y}|\mathbf{x})$, where the set of random variables $\mathbf{y}$ is called *query* variables and $\mathbf{x}$ is called the *evidence* (Koller and Friedman, 2009; Murphy, 2012). This algorithm marginalizes out any variable z that are neither query or evidence in order to compute the inference of the query variables. The mathematical notation for computing such inference is given in Equation 2.28:

$$p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y},\mathbf{x})}{p(\mathbf{x})} = \frac{\sum_z p(\mathbf{y},\mathbf{x},\mathbf{z})}{\sum_{z,x} p(\mathbf{y},\mathbf{x},\mathbf{z})} \tag{2.28}$$

As an example, consider the DAG tree in Figure 2.8. Suppose the inference problem is $p(n_o|n_i = I)$, where n_i is realized by the value I. The computation of the joint marginal $p(\mathbf{y},\mathbf{x})$ is as follows:

$$p(n_o, n_i = I) = \sum_{n_j, n_k, n_l, n_m} p(n_i = I, n_j, n_k, n_l, n_m, n_o) \tag{2.29}$$

$$p(n_o, n_i = I) = \sum_{n_j, n_k, n_l, n_m} p(n_i = I) p(n_j | n_i = I) p(n_k | n_i = I) p(n_l | n_j) p(n_m | n_j) p(n_o | n_j) \tag{2.30}$$

We can marginalize out factor by factor:

$$p(n_o, n_i = I) = \sum_{n_j, n_k, n_l} p(n_i = I) p(n_j | n_i = I) p(n_k | n_i = I) p(n_l | n_j) p(n_o | n_j) \sum_{n_m} p(n_m | n_j) \tag{2.31}$$

Where the factor $\sum_{n_m} p(n_m | n_j)$ sums up to 1. We can continue marginalizing out:

$$p(n_o, n_i = I) = \sum_{n_j} p(n_i = I) p(n_j | n_i = I) p(n_o | n_j) \sum_{n_k} p(n_k | n_i = I) \sum_{n_l} p(n_l | n_j) \tag{2.32}$$

$$p(n_o, n_i = I) = p(n_i = I) \sum_{n_j} p(n_j | n_i = I) p(n_o | n_j) \tag{2.33}$$

We can re-state the last factor from Equation 2.33 as the factor $\tau(n_o)$ which depends only on the query variable. At the end we obtain:

$$p(n_o, n_i = I) = p(n_i = I) \tau(n_o) \tag{2.34}$$

Computing the marginal probability for the evidence is now straightforward and simple. The complexity of this algorithm depends on the biggest factor of the Bayesian network. In our case, since we only deal with BN trees, every conditional probability distribution has at most four entries since each node has at most one parent.

2.4 Symbolic Models

In this section, we explain the fundamentals of a symbolic model, namely first-order Horn clauses. This type of logic rule has been widely used in the literature of artificial intelligence and data mining.[13] For example, in the construction of expert systems, or more generally, in the construction of knowledge from databases. Also, Horn clauses have been widely used in explaining predictions of black-box systems, which is precisely our target. We will use Horn clauses in order to explain the matrix factorization system from Chapter 4. Logic rules are so easy to follow when predicting new information, that they have gained a highly respected position in many communities. Thus, in this section we explain the basics we need in order to understand their role in Chapter 4; we explain fundamental aspects such as syntax and semantics, and how we can use logic rules to predict new information. We leave the learning procedure to Chapter 4, where we explain a particular algorithm to induce Horn clauses from data.

2.4.1 First-order Horn Clauses

First-order and propositional logic rules have been widely used in applications such as expert systems (Buchanan and Feigenbaum, 1978), where a set of logic rules comprises a knowledge base used by an inference algorithm in order to produce an answer to a query in a given domain. In data mining applications, such as market basket analysis, a type of logic rules called *association rules* (Agrawal et al., 1993) are mined from a database (usually a purchase one) in order to discover implications between the item sets.

First-order Horn clauses is a formal model characterized by three aspects: Syntax, semantics, and inference (Russell and Norvig, 2003). In what follows we describe each of these aspects.

2.4.1.1 Syntax

The first aspect of first-order Horn clauses is syntax, which refers to the elements of the model and the structure of the rules. The alphabet consists of operators

[13] In the data mining community, this type of model is better know as associative rules.

$(\neg, \wedge, \vee, \forall, =)$, variables, brackets, function symbols and predicate symbols. The grammar elements are *terms* (constant symbols, variables, functions) and *formulas* (composition of atomic formulas by means of connectives). An atomic formula is an n-ary predicate symbol, i.e., a relation between n arguments; for example, *cityOf(London, England)*.

A Horn clause is a universally quantified disjunction of literals (an atomic expression or its negation) of the form: $\forall x_1 \forall x_2 ... \forall x_n (\neg A_1 \vee \neg A_2 \vee ... \vee \neg A_k \vee B)$, which is equivalent to the implication $\forall x_1 \forall x_2 ... \forall x_n (A_1 \wedge A_2 \wedge ... \wedge A_k) \rightarrow B$, where $A_1, ..., A_k$ are the body of the implication and B is the head (note that both, body literals and head literal are positive literals). In this work, we restrict Horn clauses to be of the form: $\forall x_1 \forall x_2 A(x_1, x_2) \rightarrow B(x_1, x_2)$, where the clauses are range restricted (arguments of the body predicate necessarily occurs in head predicate), and the body of the clause is of size one.

As part of the syntax, the valid connectives in first-order Horn clauses are negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and implication ($\rightarrow$) (the use of these connectives is restricted to the structure of a valid clause as defined above). Unlike first-order logic where a clause can be either *universally quantified* ($\forall$), *existentially quantified* ($\exists$), or both, Horn clauses are only *universally quantified*, which means that a literal A_i having as argument any of the universally quantified variables x_j will be applied to any object that bounds x_j. An example of a Horn rule is $\forall x_1 \forall x_2$ $fatherOf(x_1, x_2) \rightarrow parentOf(x_1, x_2)$.

2.4.1.2 Semantics

The semantics of a logic statement $A_i(x_1, ..., x_n)$ is related to the *interpretation* $I(A_i)$ of such statement in a given domain D. By *interpretation* we mean a function I in D that returns 1 if the binding of the arguments of A_i, $(x_1 = a, ..., x_n = u)$, is a valid assignment in D. In other words, an interpretation function maps a) constant symbols to specific objects in D, b) n-ary predicate symbols to relations over n objects holding in D, c) function symbols to functional relations in D.

An interpretation I with variable assignment α under domain D is said to be a model M of a formula A_i if $I \models A_i$, i.e., model M *satisfies* expression A_i. In

other words, we say that a logical expression A_i is *satisfiable* if there exists a truth assignment α_i of its arguments that makes A_i true. A logical expression $A_i(x_1,...,x_n)$ is true (valid) if the interpretation of each argument $I(x_i)$ refers to an object o_i in domain D and the interpretation of the predicate symbol $I(A_i)$ maps to a relation R_i in D over all the objects o_i corresponding to the arguments x_i. A universally quantified formula $\forall xP(x)$ is true if the interpretation of the predicate $I(P)$ holds for any interpretation $I(x)$, i.e., if $I(P)$ holds in D for any object bound variable x.

2.4.1.3 Inference

Inference in Horn clauses is done by the transitive closure of ground literals, i.e., by successively applying *modus ponens* to substituted Horn clauses. Modus ponens is an inference rule of the type:

$$\frac{A(a,b),\ \forall x_1 \forall x_2 A(x_1,x_2) \rightarrow B(x_1,x_2)}{B(a,b)} \tag{2.35}$$

Where the universal quantifier is eliminated by substitution α:

$$\frac{\forall x_1 \forall x_2 A(x_1,x_2)}{A[x_1/a][x_2/b]} \tag{2.36}$$

In this way the fact $B(a,b)$ is derived from the Horn clauses in the knowledge base.

Chapter 3

Literature Review

In this chapter, we explore previous work and approaches in the literature on both the types of analysis we propose for studying ReLe systems and the studies done in the natural language processing and machine learning communities. In this way, we have three sections. The first section is devoted to *representation-level analysis* and *interpretability*, since in Chapter 4 we will use methodology from previous work on interpretability of machine learning systems and we will frame it as an analysis at the representation level according to Marr's levels of analyses of information-processing systems in cognitive science (Marr, 2010). In the second section we describe previous work on *behavioral analysis* and *robustness*, since in Chapter 5 we will use methods from the behavioral science to evaluate robustness of ReLe systems. Finally, in the third section we describe both works on *internal analysis* from neuroscience literature for inspecting abilities learned by people and works done in the NLP community to study linguistic phenomena learned by ReLe systems (such as hypernymy), and we draw a parallel between the two lines of work in order to show elements in common in both types of studies.

3.1 Representation-level Analysis and Interpretability of Black-Box Systems

In this section, we describe some types of explanations from both psychology and cognitive science used to give an account of certain cognitive phenomena. First, we describe Marr's three levels of analysis of information-processing systems (Marr,

2010), namely *computational* (first level), *representation and algorithm* (second level), and *implementation* (third level). The *representation and algorithm* level is of particular interest to this work since we will provide an equivalent of analysis in Chapter 4. Furthermore, we also describe another type of analysis widely used to give an account of how cognitive competences emerge from a complex system, namely *functional analysis*. At the end, we provide a connection between Marr's levels of analysis and functional analysis in order to better situate our work of explaining a ReLe system. Also, we describe a similar line of analysis done in the machine learning community to explain predictions of machine learning systems, namely *interpretability*. We note that we are not the first to propose to do an interpretability analysis to machine learning systems; however, we contribute with tying together interpretability analysis with representation-level analysis in Chapter 4.

3.1.1 Marr's Levels of Analysis

Marr (2010) suggested three types of analysis in order to thoroughly study information-processing systems, such as the brain and artificial intelligence systems (Marr, 1977); each of these analyses lie in different levels of abstraction. The first level, the *computational level*, is actually not intended to study any aspect of the system under analysis, but rather is intended to formulate both the principles and theory of the task that the system carries out. The second level, the *representation and algorithm level*,[1] tries to provide a model of *how* the system maps inputs to outputs; this is done via a two-fold analysis: On the one hand, one goal of this analysis is to provide a suitable representation of input and output variables for realizing the task described at the previous level; on the other hand, another goal is to be provide a plausible process, in the form of an algorithm, that describes the mapping from input to output. Finally, the third level of analysis, the *implementation level*, accounts for a description of the mechanisms (hardware) found in the system. As we can see, each of these levels has a different purpose in order to deeply understand the working of the target system, while complementing to each other.

[1] In the following, we refer to this level as representation-level analysis.

Marr (2010) proposed these levels in the hierarchy mentioned above (from first to third level) in order to better study a system.[2] Let us put in other words each of the levels to better see his point. The computational level theorizes *what* is to be computed and *why* such a theory is correct (plausible) for the task that the system executes[3] (Marr, 1977); the representation and algorithm level provides a way to represent inputs and outputs and describes *how* is the process carried out by the system to transform from a specific input to its associated output, i.e. this level proposes a suitable representation and an algorithm to carry out the theory described at the computational level; the third level describes the implementation details of the system, i.e. it accounts for the physical mechanisms that realize (implement) the algorithm from the previous level and give rise to the observed behaviors. Thus, according to David Marr in (Marr, 1977, 2010), studying an information-processing system following the order described above leads to a deep and organized way of understanding it: First, analyzing the task to be computed by the system allows us to understand both the goal and the logic of the computation (Giosuè et al., 2014); once we have characterized and formalized the task, then we are able to propose a suitable representation and an algorithm to implement the task based on behavioral data of the target system, i.e. we implement a model that carries out the target task constrained by the input-output behavior of the system;[4] finally, once we have understood what is a plausible process by which the system maps inputs to outputs, we can analyze the physical mechanisms that produce such transformations.

However, as Marr noted in (Marr, 2010), the three levels of analysis are loosely connected and the enterprise of their integration should not be taken as a *rule of thumb*; i.e., the study of a particular system may only require one or two of these

[2]The main type of explanation in neuroscience, the field David Marr pertained to, was (arguably, still is) mostly at the implementation level, ignoring any other type of explanation of the system's working that could be independent from its physical components; in other words, people tended to ignore computational and functional aspects of the systems under study, while Marr realized that an explanation exclusively at the implementation level serves more as a description of the physical arrangement of the components than as both an explanation of the function of each of the components and why the system behaves as observed in terms of its functioning objectives (Marr, 2010; John, 2015).

[3]Or the task to be executed by the system, if we are designing and implementing the system.

[4]As Marr noted in (Marr, 2010), the choice of both a representation and an algorithm may be also constrained by the physical mechanism of the system.

levels, depending on the objective of the study (usually dictated by the field's philosophy) and the level of understanding targeted. (As it will be our case in Chapter 4 where we choose to study a ReLe system only at the representation level.)

An example of how Marr's levels of analysis can be applied to study a system can be found in Marr's book *Vision* (Marr, 2010) where he proposes the study of a cash register in a supermarket. At the computational level, we analyze what is the task the machine is intended to be used for; in this example, the cash register will do sums to add up prices, thus the task is that of arithmetic. The theory behind arithmetic tells us of some properties to be considered to properly implement addition. First, a sum is commutative, which means that the order in which two elements (products' prices in this case) are added up is irrelevant for the total price, i.e. $a+b=b+a$; second, a sum is associative, which means that how we arrange the sum of some elements in an expression is also irrelevant for the total price, i.e. $a+(b+c)=(a+b)+c$; third, there exists an element which does not alter the total sum when added up, namely zero, i.e. $a+0=a$; fourth, there exists the inverse of an element such that when added up to its inverse it results in zero, i.e. $a+(-a)=0$. Besides formalizing the logic of the task, we are required to provide the reason why this particular setting is the correct one and not another one. In this case, it is easy to see that a cash register should perform sums and not, say, multiplications because the total amount to pay in a transaction at the supermarket reduces to counting all the pounds to be paid across all the products, and an addition is the operation that let us do this count; furthermore, the organization or arrangement of the products to be paid should not modify the final bill; and should a product be returned, we should get back the amount of money paid, i.e. the cash register should perform an addition using the inverse element of the price paid.

At the second level of the study of the cash register –the representation and algorithm level– we pick both a representation of the input and output of the system and an algorithm that maps input to output. In this example, it seems convenient to pick an Arabic representation and to operate upon it (any user may be well acquainted with this representation); the algorithm of the sum could be the usual pro-

cess of adding up from the rightmost digit (least significant digit) to leftmost one (most significant digit) and carry if the sum of any two digits goes equal or above 10. At this level of explanation, we can see how the task can be realized via a plausible process that manipulates a representation of input and output variables. Notice, however, that up to this point this process is independent (in this particular example) of any physical mechanism which corresponds to the third level of analysis.

In the implementation level, we seek to describe which physical device may implement the algorithm provided in the previous level. Without going to irrelevant details for the objective of our work, we can propose to implement the cash register via electronic circuits or via mechanical gears (way much complicated but possible). If the cash register happens to be already implemented, then here it is our duty to open up the system and to describe all of its elements and the way they work when given an input.

3.1.2 Functional and Mechanistic Analyses

In psychology,[5] functional analysis seeks to explain how a *capacity*[6] of a particular system leads to the system's current behavior; i.e. it seeks to explain how the system functions (Cummins, 1975; Block, 1990; Barrett, 2014; Roth and Cummins, 2014). This explanation accounts for the organization of sub-capacities of the system that when properly assembled give rise to the target capacity. For example, in order to explain how long-term memory works, it is necessary to describe it in terms of its sub-components, namely memory encoding, storage, and retrieval (Barrett, 2014). The analysis will describe how long-term memory processes information by virtue of the organization of its sub-components. This analysis is taken as a type of *functional explanation*; the explanation obtained describes a cognitive process for which we do not need to involve any parts of the brain, such as neurons; rather the explanation is based on capacities. This explanation, thus, is able to provide an account of behavioral data collected when the phenomenon under consideration

[5] As well as in other disciplines such as biology or engineering (Cummins, 1975; Roth and Cummins, 2014).

[6] Also called *disposition* (Cummins, 1975), *competence* (Block, 1990) ,or *effect* (Roth and Cummins, 2014).

occurs.

In general, any functional analysis of a system can be represented as a block-diagram (also called box-arrow diagram) or as a flow-diagram (Roth and Cummins, 2014). In this model, each component within the system is ascribed with a function, i.e. this component has a well-defined structure, with clear boundaries, that allows it to perform a well-defined input-output process. The objective of this model is to show how the information is processed by the system in order to produce the target capacity, i.e. it shows how the information goes through a certain path, or flow, and not through other possible paths in the state space of possible configurations (Roth and Cummins, 2014). A toy example of a functional analysis is the explanation of how a multiplier works taken from (Block, 1990). Let us suppose that the algorithm for carrying out a multiplication $a \times b$ is to simply add b to itself a times; thus, we can propose the following sub-components of the system. First, we propose a counter that starts from 0 and increments by 1 each time-step; also, we propose a component that checks whether the counter has arrived to the value of a, in which case a halter will stop the process; if the counter has not arrived to a then an adder will sum the value b to itself. The organization of these components is almost fully described; at each step, the checker will check if the counter has arrived to the number a, if so the checker sends a signal to the halter to stop; otherwise, the adder sums the number b to itself and the counter proceeds to count. This is how *multiplication* functionally arises from a set of components. Thus, a functional analysis allows us to see why the information is processed as such, i.e. why we observe the behavior we observe in terms of the internal functioning of the system as a decomposition of its internal sub-capacities.

From a further inspection of the above example, we notice two things in the analysis. First, we do not provide any physical consideration of how each functional component of the system can be implemented (through mechanical gears, or electronic circuits, or biological means) as we only care about the functions that each component carries out and the organization of such components. Second, we see that each component of the multiplier may be a complex system, such as the adder;

depending on the complexity of a component, it may be a candidate to have in its own right a functional analysis in order to explain its own capacity. This situation leads to questioning what are the boundaries of a functional analysis, i.e. what is the correct level of abstraction sought in such an analysis? In some cases, even though the answer may depend on the objectives of the researcher, in general, the boundary of a functional analysis lies when a *primitive processor* is reached out in the explanation of the target system (Block, 1990). A primitive processor is an artifact whose working is no longer a matter for the psychologist to explain; i.e., a primitive processor is a *low-level* component that when assembled, or organized with, other primitive processors realizes (implements) a capacity. Therefore, a primitive processor is to be found at the lowest-level of analysis, namely the implementation level (Marr, 2010) (as we saw in Section 3.1.1) or the *mechanistic level* (Kaplan, 2017); in this level, a psychologist is no longer required to provide any sort of description, or explanation, of how the component works since such a component is no longer ascribed with any capacity, but rather it is ascribed with a mechanistic functioning.[7] For example, we could further explain the capacity of the adder to perform additions, and we would do this by further decomposing this component into sub-components, and we could proceed to decompose further sub-components until we reach out an elementary circuit, such as an *and* gate whose inputs are two voltages and its output is a single voltage;[8] to this primitive component, we attach no cognitive capacity since it does not carries out a *high-level* function, i.e. it does not perform any *meaningful* competence.

As we have seen, functional analysis provides an abstract explanation, in terms of functions, of how a system works. As such, it is independent of the hardware that may implement the system under study. Also, we saw that primitive processors lie in the hardware level; these elementary components implement the system and thus

[7]This mechanistic function, as already mentioned, is not to be understood as a function in the sense of a cognitive capacity, but rather as an operation that by itself has no *semantic* interpretation (Block, 1990). However, it may be the case (though possibly unlikely) that a single primitive processor accounts for the whole capacity of a component, according to the analysis proposed by the researcher.

[8]This circuit simply outputs around 7 volts if both inputs are the same level of voltage, or around 4 volts if the inputs have different levels of voltage.

give rise to the observed behavior. But, if a psychologist works at the functional level, who is then responsible to explain the organization and working of the primitive processors? Are functional explanations *truly* independent from the physical level? Is a functional explanation self-contained in its own right? These philosophical questions have been addressed by both philosophers and psychologists (Barrett, 2014; Kaplan, 2017), and there are advocates of different sorts of theories. Before succinctly describing these theories, we answer the first and easiest question; there are disciplines whose objective is mainly to characterize, study, and describe physical components, such as neuroscience or physiology. These disciplines, in contrast to psychology, aim to discover the properties, functions, and organization of what we have called primitive processors, which are defined according to the theories and laws in such disciplines.[9] Thus, an explanation in a field such as neuroscience aims to describe the physical mechanism in a brain that gives raise to a behavior Barrett (2014); this low-level explanation is not concerned with ascribing a high-level function to a component (or group of components) under study, such as ascribing a *hand detection* function to a group of neurons in the part of the brain responsible for vision in a human.[10] Thus, this type of explanation concerned only with descriptions of physical entities is called *mechanistic explanation*. With regard to the second and third questions from above, there are opposing views, as well as integrative views, on the autonomy of functional analysis from mechanistic analysis. For example, there is a widely-accepted theory among psychologists (and rejected by some neuroscientists) that claims that a functional analysis is independent (autonomous) from a mechanistic analysis; i.e. explaining the capacity of a system can be done without knowing anything about its hardware (Barrett, 2014; Kaplan, 2017). On the other hand, there is the view that a functional analysis is rather a complement to a mechanistic analysis, i.e. it serves as a first approach towards understanding the physical mechanism of the target system, and when both functional and mechanistic analyses are provided then a complete (multi-level) explanation of the system is obtained

[9] For example, a primitive component in neuroscience is a neuron.

[10] Even though this claim is true in many neuroscience research programs, is not true in fields such as cognitive neuroscience where the objective is to actually discover functional properties (capacities) encoded in groups of neurons (Geary, 2005).

(Piccinini and Craver, 2011; Kaplan, 2017). However, given that our objective is not to comprehensively describe these views, but to rather describe the role of a functional analysis in explaining a system, we leave the description of these view as future work.

3.1.3 On a Comparative View Between Marr's Levels of Analysis and Functional-Mechanistic Analyses

In order to better situate our work in Chapter 4 in the grounds of psychology and cognitive science, we aim to provide a succinct view on the relationship between Marr's levels of analysis of information-processing systems and the explanations of complex systems obtained via functional analysis.

As we saw in Section 3.1.1, Marr structures the analysis of systems into a hierarchy of three levels, namely the computational level, the representation and algorithm level, and the implementation level. The first level is targeted to define and describe the theory of the task to be computed by the system; in the second level, the researcher proposes both a plausible way of abstracting –representing– the inputs and outputs and a model –process– of how an input is mapped to an output; the third level is a description of the whole physical device that implements the system in a component by component basis and their relationships. On the other hand, we saw in Section 3.1.2 that a functional analysis provides an explanation of a target capacity of a system in terms of the sub-capacities of the system's components; however, this explanation is totally independent of any physical artifact, since it concerns only the high-level functioning of the system, not the mechanical working, which is left for a mechanistic analysis.

At a first glance, there is a clear correspondence between the objectives of Marr's implementation level of analysis and mechanistic analysis. Both analyses seek to understand how the low-level components of a system give rise to the system's behavior. Thus, the explanations obtained via the two analyses correspond to the description of the mechanistic working of the physical devices that compose the system.

There also seems to be a correspondence between the other two levels of anal-

ysis proposed by Marr and functional analysis, since apparently these types of analysis work independently of the system's hardware; besides, these analyses try to explain, in a high-level view, how the system functions, i.e. they try to account for how the system processes information. However, in a deeper inspection, we claim that there is no bidirectional correspondence between Marr's first two levels of analysis and functional analysis, but rather an unidirectional correspondence: On the one hand, every functional analysis corresponds to an analysis at the representation and algorithm level; on the other hand, an analysis at the representation and algorithm level corresponds to a functional analysis if and only if it fulfills the following requirements. A functional explanation provides the design of a system in terms of sub-components where each of them is ascribed with a particular function; this functional design can be easily put in the form of an algorithm along with an appropriate representation, and as such is able to explain how the system processes the information from input to output while explaining a target capacity (language, vision, etc.). However, an algorithm that explains the input-output process of a system is not necessarily equivalent to the system's design provided by a functional explanation (Roth and Cummins, 2014); an algorithm shows a process that not necessarily explains a capacity of the system under study via a decomposition into its sub-components while explicitly accounting for the sub-capacities encoded in such a system (as it is our case in Chapter 4.) Put under another view, if and only if we can deduce the functional design of a system's capacity via a proposed algorithm coming from Marr's second level of analysis, then we claim that such algorithm accounts as a functional analysis.

Finally, we claim that an analysis at the computational level is outside the scope of a functional analysis, and thus is not to be expected in a functional explanation. An analysis at the computational level describes the task to be computed by a system and provides an account of the theory and principles of such a task; thus, this analysis does not involve any aspect of the system, whether functional, algorithmic, or mechanistic. On the other hand, as explained before, a functional analysis is a functional account of a system's behavior with regard a capacity; as such, a func-

tional explanation does not aim to explain the theory and principles that the system is implementing or realizing, but rather it aims to explain the functional properties of the system.

Overall, there is a difference between Marr's framework of explanation and the functional-mechanistic framework of analysis which stems mainly from the background where the two frameworks originated from. David Marr, as a neuroscientist,[11] seemed to be more concerned with the practical use of an explanation rather than with any philosophical implication (John, 2015; Kaplan, 2017); thus, under this view and leaving aside some philosophical matters,[12] Marr (2010) suggested some relationships among his three levels of analysis, as we mentioned in Section 3.1.1, where the most significant relationship for our discussion is that between the representation level and the implementation level. Marr (2010) argued for the implementation level to possibly constrain the choice of algorithm and representation; i.e., not any possible algorithm and representation are good candidates to explain the input-output process carried out by a system, unless the system's hardware is able to execute such algorithm. For example, a system whose hardware runs in serial mode would not be able to implement an algorithm that runs in parallel. Thus, the choice of algorithm and representation may be influenced by physical constraints from the system's hardware. On the other hand, some advocates of functional analysis (in particular philosophers) put much emphasis on the philosophical implications of this type of analysis (Cummins, 1975; Barrett, 2014). These advocates defend the view that functional analysis is totally autonomous from any mechanistic analysis (as oppose to Marr's view with regard to his levels of analysis.) Thus, if a proposed model obtained via functional analysis is able to explain behavioral data,

[11] Some neuroscientists seem to be agree with David Marr that in an explanation of an information-processing system, the algorithmic level is not fully independent from the implementation level, but complementary (Smith and Kosslyn, 2007).

[12] Marr briefly mentions the argument of *multiple realizability* in (Marr, 2010), though he did not elaborate on this argument as much as some advocates of functional analysis have done (Barrett, 2014; Kaplan, 2017). Multiple realizability is an argument in favor of the autonomy of functional analysis with respect to mechanistic analysis; it claims that the same input-output behavior can be observed from different types of information-processing systems where the difference lies on their physical implementation. Therefore, if all the systems exhibit the same behavior, this means that it is possible to model the functional working of any of these systems independently of their hardware, and thus a functional explanation serves as a faithful model of any of these systems.

then this model is accepted as a valid explanation Kaplan (2017); Barrett (2014), independently of the physical mechanism that gave rise to such data. Therefore, even though there is a clear correspondence between the analysis done at the implementation level and the analysis carried out by a mechanistic analysis, these two types of analysis differ in a philosophical aspect, namely how people embodies them within an explanatory framework, either as a complement to constrain the other levels of analysis or as an independent type of explanation.

3.1.4 Interpretability

Interpretability characterizes the degree to which machine learning (ML) systems are understandable for a human: If a person is able to understand the *reasons* why an ML system predicted an output y_i, then we deem such a system as interpretable. By the term *reasons* we mean the relationship among a subset of independent variables $x_i, x_j, ..., x_m$ from the input domain and the output (dependent) variable y_i. For example, if the task is that of classifying if a house is *cheap* or *expensive* based on its attributes (location, number of rooms, when it was built, and so on) we could build a classifier (a type of ML system) that either is interpretable, and thus we would be able to know that a house is cheap because it has less than 3 rooms, it is older than 50 years, and it is located in neighbourhood *XYZ*; but we could also build a classifier that is a *black-box* where the output is related to the attributes in unclear ways for us.

An example of an interpretable ML model is a classifier that linearly relates input and output variables, for example a logistic regression classifier (Section 2.2.1). This model is considered to be understandable since it is clear to see the weighted contribution (influence) of each independent variable to the final decision. Whenever an interpretable relation among input and output variables is given by the structure and/or the parameters of the ML model, we consider such a relation to be a type of *explanation* of the predictive behavior of the classifier. We note that an interpretable ML model does not necessarily needs to be linear. Other common types of interpretable models are decision trees and logic rules.

The importance of interpretability has been widely acknowledged in the ma-

chine learning community as a way of validating the knowledge learned by the model (Taylor and Darrah, 2005; Andrews et al., 1995; Tickle et al., 1998; Ribeiro et al., 2016). In addition, in practical domains where machine learning models are deployed into systems to be used by a user non-expert in machine learning, it is important to have an explanation of each prediction; this explanation serves as a support of the system's decision that provides the user with confidence of the system (Druzdzel, 1996). Domains where explanations of ML system's predictions are sought are medicine (Wall et al., 2003; Kim et al., 2006; Lisboa et al., 2008), finance (Baesens et al., 2003; Verbeke et al., 2011), and biology (Liu et al., 2014), among others.

An explanation not only serves for understanding the predictions of an ML system, it may also fit other objectives. For example, verifying the knowledge learned by the ML system, or as a way for understanding what phenomena the system has captured from the dataset. For example, suppose the problem of predicting the presence of a disease based on the symptoms of a person. Both the disease and the symptoms are grounded in output and input variables y and $x_1, ..., x_n$, respectively. Having an interpretable model that explains how the symptoms relate with the disease may shed light on new relationships previously unknown, which in turn may help to build new symptom-disease theories.

However, a large portion of interpretable systems are linear, which fail to accurately fit high-dimensional datasets. Notable exceptions, in specific domains, are the works of Caruana et al. (2015) and Letham et al. (2015); nevertheless, it is unclear to what extent linear models can scale to other domains (higher dimensional or more complex.) A solution to accurately learn high-dimensional datasets, is to use more complex[13] ML models, such as neural networks, support vector machines, and matrix factorizations.

Complex ML models have been widely used for classification tasks due to their ability to learn high-dimensional datasets, and thus accurately separating instances from different classes. However, due to their complex nature, it is difficult to ex-

[13]We define the complexity of a model in terms of either its size (number of parameters) or the relation among parameters (non linear.)

plain a prediction[14] (d'Avila Garcez et al., 2001; Choi et al., 2016). These models can be seen as *black-boxes*,[15] because their structure and parameters, per se, do not provide an understandable explanation of how the inputs relate to the output; this happens because either the input data is transformed using non-linear functions and, therefore, its relation with the output variable becomes intricate, or the size of the model prohibits a clear and simple analysis of the input-output relation. A classic example in the literature of a black-box is a neural network (NN), where the input data is distributed among neurons (processing units) which linearly combine it and non-linearly transform it.[16] Therefore, understanding why an NN predicted an output y_i, in terms of its intricate internal machinery, is difficult for a human.

Previous work in the machine learning community has tackled the problem of providing explanations for the predictions of black-box models (Baehrens et al., 2010). The main idea is to learn a *proxy* or *descriptive* model, that is interpretable and mimics the predictive behavior of the black-box model; this means that the proxy model encodes in its parameters and structure, to some extent, the knowledge learned by the complex model. How faithfully can the proxy model capture the knowledge of the black-box model? This is a natural question that arises up to this point, and the answer may depend on several factors: The form of the black-box model, the selection of the proxy model, the approach to learn the proxy model, the scope of explanation sought for a prediction, and the way to operationalize *fidelity*.[17]

In the following sections, we describe how previous works have tackled the problem of interpretability by looking at it from three main angles: The choice of descriptive model, the approach for learning the descriptive model, and the type of explanation sought. In Chapter 4 we will provide a critical appraisal of how previous work may or may not be helpful to understand the decision process of the

[14] Some of moderately complex models are easier to understand, such as decision trees.

[15] We will indiscriminately use the concepts complex and black-box to refer to a machine learning model, or system, for which is difficult to understand its predictions and the knowledge it has learned.

[16] Except when no hidden layers are present, and the neurons do not have non-linear functions (except for the output neuron), in which case the NN is analogous to a logistic regression model.

[17] This concept is commonly operationalized as how close is the predictive behavior of the proxy model to that of the complex model in terms of an accuracy metric. We will show in Chapter 4 that the choice of this metric is crucial for a proper evaluation of fidelity.

ReLe system we aim to analyze.

3.1.4.1 Learning Interpretable Proxy Models: Decompositional vs. Pedagogical Approaches

Learning an interpretable proxy model from a black-box model is often termed *knowledge extraction* since it is intended that all the knowledge encoded in the complex model is transferred to the proxy model. There are two main approaches for learning a proxy model: *Decompositional* (Towell and Shavlik, 1993) and *pedagogical* (Craven and Shavlik, 1995). In a decompositional setting, the internal machinery –parameters and structure– of the black-box model is inspected in order to build an interpretation of the logic of how the complex model makes a prediction (Martens et al., 2009). In this setting, the black-box model is not really treated as such since the functioning of each internal component is analyzed. And sometimes a thorough delineation of the internal components is provided (d'Avila Garcez et al., 2001). In contrast, in a pedagogical approach, the complex system is seen completely as a black-box and the way to extract its knowledge is to treat it as an *oracle* by observing its input-output behavior (Thrun, 1994).[18]

An example of a decompositional approach, in the ML community, to learn a proxy model for ReLe systems is the work of (Yang et al., 2015). In this work, sets of horn rules are extracted from two tensor factorization systems. The form of each rule is $B_1(a,b) \wedge B_2(b,c) \implies H(a,c)$ where predicates B_1, B_2, and H are associated to relations over entity types, and arguments a, b, and c are associated to types of entities; therefore, each predicate and argument is represented by an embedding. An example is the rule $BornIn(person_A, city_B) \wedge CityOf(city_B, country_C) \implies Nationality(person_A, country_C)$. The method to construct the logic rules is to search, in relation space, for relations such that the form of the rule described above is fulfilled.[19] This search may lead to several candidate rules; then, in order to prune candidates, the predicates in the body of each candidate rule are composed into a single vector (the embeddings are summed or multiplied), and a k-nearest neighbor

[18] These two approaches have been mainly used in the machine learning community.

[19] Yang et al. (2015) take advantage of the fact that the entities are typed, which reduces the search space considerably.

algorithm selects the closest (and most relevant) rules to the head predicate H. At the end, the remaining set of rules comprise the knowledge encoded in the tensor factorization systems.

In general, decompositional methods are *ad hoc* to the black-box model, which makes it hard to extrapolate to other models. And most of the previous work is applied to neural networks for machine learning problems (Vaughn, 1999; Tsopze et al., 2011; Murdoch and Szlam, 2017; Odajima et al., 2008). For these reasons, a more generic approach –*pedagogical*– is sometimes preferred.

The pedagogical approach sees the complex model as a black-box and has no intention of inspecting any of its internal components (Saad and Wunsch, 2007; Domingos, 1998). The methodology is to observe input-output relations of the black-box and to use these observations to learn the interpretable proxy model. More concretely, the black-box model is used as an *oracle* in order to re-label instances from a dataset:[20] For each input instance $\mathbf{x}_i$, the black-box produces a class label $\hat{y}_i$. These new pair $(\mathbf{x}_i, \hat{y}_i)$ becomes a training instance for learning the proxy model. The objective is to have the proxy model behaving like the black-box, i.e. mimicking its predictive behavior. Learning the proxy model via a training set avoids the need of taking into consideration the internal mechanisms of the black-box model; therefore, this approach is easier to extrapolate to a wider set of models than the decompositional approach.

An example of a pedagogical approach in the ML community is the work of Craven and Shavlik (1995). The target black-box system is a neural network classifier which is given input data in order to produce labels and build a training set for a proxy model. Then, decision trees are induced using the training set derived from the black-box system. This proxy model accurately mimics the predictive behavior of the neural network and thus serves to explain its predictions. We note that an advantage of the pedagogical approach, as it is used in the work of (Craven and Shavlik, 1995), is the possibility of manipulating the input data to produce more instances to be labeled by the neural network.

[20] This dataset is usually the same used for training the black-box model.

3.1.4.2 Explanations: Global vs. Local

There are two types of explanations in terms of their scope: Global and local. In a global explanation, a proxy model captures all the knowledge encoded in the black-box model; therefore, the proxy model is able to explain any prediction. On the other hand, a local explanation only serves to explain a particular prediction, i.e. the proxy model is learned just for a single input-output instance $(\mathbf{x}_i, y_i)$.[21]

Each approach has its own advantages and disadvantages. The main advantage of learning a global interpretable model is two-fold: We train once a model that explains any prediction, and we are able to observe in a cohesive way all the knowledge encoded in the black-box model. The advantage of learning local explanations is training a simpler proxy model, since the number of training instances is considerably fewer than those required for learning a global explanation. On the other hand, obtaining a global proxy model that faithfully resembles the complex model may be unfeasible, or even NP-hard (Jacobsson, 2005), depending on the selection of proxy model. But, if we aim to have a complete picture of the logic underlying the black-box model, then a local model may fail to do so.

Two relevant examples are the works of Craven and Shavlik (1995) and Ribeiro et al. (2016) (from the machine learning and data mining communities.) First, Craven and Shavlik (1995) build decision trees as global proxy models for neural networks. The method used for learning the proxy models is pedagogical, the NNs re-labels training instances. The decision trees faithfully capture the knowledge of the NNs in four different domains, while generalizing to unseen instances almost as accurately as the NNs itself. Second, Ribeiro et al. (2016) build linear models for explaining single predictions $\hat{y}_i$. In order to do this in a pedagogical way, a perturbation is applied to the input instance corresponding to the prediction to be explained, namely $\mathbf{x}_i$. This perturbation consists of altering a feature value x_j of the instance. After that, a new instance $\mathbf{x}_k$, derived from the perturbation, is given as input to the black-box to produce a label. In this way, a new training instance

[21] Even though the proxy model serves to explain only a single prediction, it is necessary to obtain a sample of instances in the neighborhood of $\mathbf{x}_i$ in order to build a training set for learning the interpretable proxy model.

$(\mathbf{x}_k, \hat{y}_k)$ is obtained in order to learn the proxy model. This procedure is applied several times in order to get a training set of the desired size. Depending on the objectives of a research project, a global explanation may be preferred over a local one since it gives a wider view of the black-box system under analysis.

3.1.4.3 Selection of Interpretable Proxy Model

Previous work in the ML and AI communities has proposed a variety of interpretable proxy models. While the most common model is logic rules (Thrun, 1994; Setiono et al., 2009; Lehmann et al., 2010; d'Avila Garcez et al., 2001), other models have been used, such as decision trees (Craven and Shavlik, 1995), linear models (Ribeiro et al., 2016), and state machines (Jacobsson et al., 2007). Logic rules have been used mainly in decompositional approaches for neural networks, since describing input-output relations in neuron units can be easily done with a logic rule. The rest of the models have been used in pedagogical approaches, since learning them from data produced by the black-box model reduces to a usual machine learning setting.

Several factors are considered when choosing a proxy model. The first factor is the complexity of training. Learning logic rules of arbitrary size from data is NP-hard; therefore, learning them requires heuristic approaches that may impact on the fidelity.[22] A second factor is the ability of the proxy model to serve as a global explanation in the case where the black-box system outputs a vector instead of a single value (a multi-label prediction problem (Murphy, 2012)). Logic rules and state machines are able to fit this purpose, but other models, such as decision trees and linear models, are able to fit only a single target variable; therefore, it is required to learn one decision tree, or one linear model, for each of the output variables in the black-box system.

A third factor in consideration is the expressiveness of the proxy model to faithfully capture the knowledge of the complex model. Logic rules, decision trees, and state machines faithfully capture small neural networks (usually, the input size

[22] In the case of state machines, it is unclear the overall complexity of learning as claimed in (Jacobsson, 2005).

is less than a 100 features), but it is unclear to what extent they can generalize to bigger input spaces. The last aspect to consider is how comprehensible (understandable) the proxy model is for people. Huysmans et al. (2011) provide evidence towards small[23] graphical proxy models being easier to understand than both non-graphical ones, such as logic rules, and bigger graphical models, such as decision trees or state machines.

3.1.4.4 Other Approaches Of Interpretability

There are other approaches in the literature to understand predictions of black-box models: Visualizations and pattern extraction. The main similarity between these two approaches is the form of the explanation; in both cases, meaningful patterns that give a hint of either how the complex system is working or what are the most important variables for the prediction are given to the user. The main difference with the approaches previously described is the definition of explanation. While previous works in interpretability seek to recover a structure that resembles the logic of the inner working of the black-box system, these two lines of work only provide patterns, such as activation of neurons (for NNs), distance between representations learned (for ReLe systems), or words (for NLP systems), that show statistical regularities but lack any sort of structure.

For example, in the work of Lei et al. (2016) (from the NLP community), in order to explain a prediction, the set of input patterns most relevant for the output are extracted. Lei et al. (2016) evaluate this approach on the task of sentiment analysis, where input patterns correspond to words in a text (a review of a product from a user), and the prediction is the sentiment of the user (towards the product). In that work, no interpretable model is learned, but rather a bag of features is extracted. This approach does not account for a structured explanation of how each of the features is related to the output.

Visualizations are another method for either obtaining a rationale for a single prediction, visualizing the phenomena that has been captured by the black-box model in its internal machinery, or measuring how close are the representations of

[23] In the number of parameters.

objects in a semantic space. Work in the computer vision community is the best example for the first case, where a saliency or heat map is obtained in order to explain a prediction (Lapuschkin et al., 2016; Zintgraf et al., 2017; Selvaraju et al., 2016). In this map, the pixels that contribute the most to the prediction are highlighted with brighter colors than those pixels that contribute less or do not contribute. In this way, it is visually easy to recognize which pixels are the most influential in the prediction.

Visualizations, as said before, also serve to show what neural networks have captured by visualizing the activation of neurons in different layers. When this visualization is applied to image classifiers, it is possible to observe some kind of building-block features that the NNs *disentangle* from the input images, as it is done by works in the machine learning and computer vision communities (Yosinski et al., 2015; Bau et al., 2017; Simonyan et al., 2013; Zeiler and Fergus, 2014), for example, edges, or basic shapes. When the visualization is applied to NLP classifiers, phenomena particular to the discourse structure can be extracted; for example, Karpathy et al. (2016) discover that a specific unit of an NN, trained on text, keeps track of the length of a phrase while generating it, while other units keep track of other discourse referents, such as quotation marks, and opening and closing parenthesis for source code. In the NLP work of Li et al. (2016), it is displayed how much a unit, in an NN, contributes to the composition of a sentence. Another type of visualization is introduced in the ML work of Van der Maaten and Hinton (2008), namely tSNE, where semantically related embeddings are clustered in a sub-region of the 2-dimensional Euclidean space. In this way, it is possible to visualize how close related objects are to each other; for example, in an NLP application, such as sentiment analysis, it is expected that the embedding of the concept *terrible* is close to that of *negative* and distant to the embedding of the word *lovely*.

A disadvantage of some visualization methods is their decompositional character. These methods are not likely to be easily extrapolated to other models due to the difference in architectures. Besides, these methods provide explanations for single predictions. Compared to explanations provided by symbolic models such as

logic rules, state machines, or decision trees, the explanations obtained from visualizations do not contain a chain of logical steps, one step derived from another, where the last step is the prediction of the black-box system; in such a multi-step explanation, a series of possible reasons for a prediction are logically assembled. Some visualization methods may account for a multi-reason explanation but no logical structure is provided.

3.2 Behavior Analysis and Evaluation of Robustness

In this section, we focus on the methodology and analyses from the behavioral science that seek to understand the behavior of animals and people in terms of environmental and internal factors, namely *behavioral analysis*. We also describe analyses done in the natural language processing and machine learning communities that seek to understand how the behavior of machine learning systems is affected by external factors, such as biases in the data; thus, these analyses seek to evaluate the *robustness* of the systems. To the best of our knowledge, we are the first to use behavior analysis from the behavioral science to study ReLe systems (see Chapter 5).

3.2.1 Behavior Analysis

Behavior analysis, a type of study from the behavioral science, seeks to provide an account of the role that different factors may play in the response (behavior) of subjects under study; these factors are treated as independent variables while the response is the dependent variable. Accounting for such influence is usually portrayed in the form of correlation measures. In order to ensure *internal validity*, the experiments are performed under controlled conditions; i.e., in order to validate that changes in the dependent variable are only due to changes in the independent variable under study, the experimenter has to isolate the former factor from possible confounding factors (Epling and Pierce, 1986). When an isolation from possible confounding factors is not possible, then a statistical control is applied in order to analyze the effect of such factors on the dependent variable (McDonald, 2014).

Besides internal validity, *external validity* is also desirable. This concept refers

to the extent to which results can be extrapolated to other groups of subjects different from the subjects studied. Usually, there is a trade-off between internal and external validity. While internal validity is fully achieved under laboratory conditions where all (or most of) the possible confounding factors can be controlled, external validity can be achieved only through real-world experiments where the subject behaves in its own environment with no restrictions imposed by the experiment. This type of study aligns to a type of observational study where the subject under study keeps his behavioral patterns unchanged and an observer collects behavioral data.

In a comparative view between experimental frameworks seeking for internal or external validity, we note that data collected from laboratory experiments may not be representative of real-world conditions, and data collected from real-world experiments may be plagued of confounding factors. Although achieving a balance between the two types of validity is an open problem, there are previous suggestions. For example, Epling and Pierce (1986) suggests to address applied research questions, such as human aggression, in order to motivate the required change in the laboratory experimental setup that can throw results generalizable to conditions outside the laboratory.

In the following sections we explore both types of experimental frameworks since both are relevant to our study in Chapter 5. We focus mainly in experiments from animal behavior since these type of studies have a long tradition in the behavioral science and their methods have shown to be rigorous and well designed (Smith and Kosslyn, 2007); also, experiments conducted on animals are non-verbal, as opposed to experiments on humans whom can verbally reply any inquiry from the researcher, which means that the researcher has to design ways of ensuring that the behavior observed was due to the target stimulus and not due to a confounding factor;[24] this experimental setup is very similar to ours since ReLe systems have no way of producing verbal reports.

[24]Though some of the experimental frameworks applied to study animal behavior are similar to the frameworks used to study human behavior.

3.2.1.1 Animal Behavior Analysis

Animal behavior may be caused by several factors, such as genetic, physiological, environmental, and psychological (Hager, 2010; Dawkins, 2003). As Hager (2010) exemplifies, ants who were born with the role of defending their colony, show the predisposition to be aggressive. On the other hand, stress in animals can be a cause to release corticosteroids, chemicals that suppress some behaviors and promotes others such as escaping from a predator. Environment, also plays an important role in the behaviors of animals; for example, an environment where there is plenty of resources may cause an animal to not to fight for food. However, investigating exactly and precisely why an animal behaves as observed from a genetic, physiological, environmental, and psychological perspectives is such a difficult endeavour. Thus, different disciplines study the effects of different factors on behavior. We devote this section to describe the experimental methods used to study animal behavior from both environmental and psychological perspectives.[25]

We start our review of previous work with two works that study psychological processes of animals through behavior analysis, namely the works of Hampton (2001) and Loukola et al. (2017). These two works possess similar experimental frameworks, where the subjects under study are placed in a laboratory in order to control for possible confounding factors that may account for the observed behavior. The researchers seek to understand if the observed behavior is due to a psychological capacity of the animals when these are engaged in a task, or is due to a confounding factor. Therefore, these two works seek for internal validity of the experiments and do not care about the natural environment of the animals.

Hampton (2001) investigated whether rhesus macaque monkeys knew if they remembered a stimulus; that is, the objective of the study was to figure out if rhesus monkeys were aware (conscious) of their recent memories. To investigate this, Hampton (2001) designed a controlled experiment in a laboratory with two rhesus monkeys as subjects under study. The experimental setup was as follows: Each monkey sat down in front of a touch screen where a stimulus appeared, in this case

[25]The literature on this subject is vast, and thus we focus only on a set of works that we believe to be representative of the field and to be related to our work.

the stimulus was a random image from a collection of images; the image was displayed for a certain amount of time and then it was removed from the screen. After an interval of time, the screen displayed two options for the monkey to select one of them; the first option was a button inviting the monkey to do a memory test where he has to remember the image displayed at the beginning,[26] and the second option was a button allowing the monkey to reject the memory test. If the monkey chooses to reject the test, a tiny reward is given to him. On the other hand, if the monkey chooses to do the memory test and fails, no reward is given, but if he succeeds then a significantly good reward is given. This experiment was repeated several times in a single session.[27] This experiment design tests for the awareness capacity of the monkeys, via behavioral tests, by encouraging them to do a memory test if the monkeys are sure that they will be able to remember the stimulus shown at the beginning of the test. In the case that the monkeys are aware that they will not be able to remember the stimulus, then it is expected that they will reject the test. In order to control for a possible unadvertised behavior of the monkeys, such as a lazy behavior where they chose the easiest option (reject the test) despite remembering the stimulus, a forced test was randomly introduced from time to time where the monkeys where not given the option to reject the test and they had to do the memory test.

In the experiments, Hampton (2001) allowed for a long time interval between the appearance of the stimulus and the choice of accept or reject the memory test such that the monkeys were likely to forget the stimulus and thus reject the test if they were aware that they could not remember the image. When comparing the results, in terms of accuracy scores, of the memory tests when the monkeys chose to do the tests against when they were forced to do them, the results indicated a substantially higher mean accuracy[28] on freely-chosen memory tests. This result seems to indicate that when monkeys knew that they were able to remember the

[26] In this memory test, four randomly-placed images are displayed on the screen and the monkey has to select the image shown at the beginning of the experiment. The monkeys were pre-trained to do this type of test.

[27] The monkeys were tested for 10 sessions.

[28] Paired t-tests on the mean of the accuracy scores of each monkey were computed. Accuracy scores of both monkeys on freely-chosen tests were around 84%, and accuracy scores of both monkeys on forced-tests were around 70%.

stimulus, they consequently chose to do the test, thus effectively rejecting the test when they knew they could not remember the image.

One possible confounding factor causing the monkeys to reject the tests, however, may be some spurious stimulus from the environment (such as a noise), or probably some internal factors (for example, a lack of motivation). Thus, in order to control for these possible confounding factors, in a new series of experiments, Hampton (2001) introduced a slightly modified test where the screen displayed no image in order to simulate a forgotten image; that is, the hypothesis was that monkeys would treat the lack of an image as a stimulus that could not be remembered, and thus every time this test was presented to the monkeys, they would reject the test. This modified test was randomly intermixed with the normal memory tests as described above.[29] Surprisingly, the monkeys rejected the modified test significantly more often than normal memory tests.[30] This systematic behavior of the monkeys shows evidence that the monkeys were aware that they did not have the stimulus on their memory, and thus they would not be able to pass the memory test.

Finally, in order to measure the effect of the time interval between the appearance of the stimulus (image) and the choice of doing or not doing the test, Hampton (2001) varied this factor from small intervals of time to large ones. As expected, both monkeys rejected significantly more often those tests occurring after large time intervals than those tests that occurred after short time intervals. This figure was also observed on forced tests but in the form of successful outcomes; the monkeys performed significantly better at those tests that were forced after a short time interval than at those forced tests occurring at large time intervals.[31] These last experiments also throw evidence, via behavioral tests, towards the hypothesis that rhesus monkeys were aware (conscious) of the items present in their memories, and they behaved as such in most of the tests.

We now describe the work of Loukola et al. (2017), which has a similar experimental framework as that of Hampton (2001), namely the subjects under study

[29] From a session of 96 tests, 10 of these were the modified memory test.

[30] A paired t-test was computed for each monkey between the mean accuracy of normal tests and the mean accuracy of modified tests.

[31] F-tests were computed for each monkey to compare accuracy means.

are situated on a controlled environment in order to see the effect of an independent variable on the behavior of the subjects. Furthermore, in the work of Loukola et al. (2017) the experiments also try to test cognitive capacities of animals. More precisely, Loukola et al. (2017) tested the cognitive and behavioral flexibility of bumblebees in an artificial task using an artificial object. A set of experiment bees was trained to move a ball from the edge of a platform (two platforms of different sizes were used) to the center of the platform.[32] When a bee failed to carry out the task, the experimenter showed the bee how to move the ball using an artificial bee moved using a transparent stick; after this demonstration, the bee would obtain the reward. A group of control bees[33] were given the same task; however, when a control bee failed a trial, the experimenter did not show how to solve the task, but provided the bee with a reward. On a test phase consisting of 10 trials, both groups were tested on the same task using the biggest platform. The results unmistakably show that experiment bees were better able to perform the task than the control bees.[34]

On another round of experiments, Loukola et al. (2017) investigated the effect of different types of learning on bees, namely social learning, *ghost* learning, and no demonstration. To do so, a group of bees was pre-trained to move a ball from the edge of a platform[35] as in the experiment described in the paragraph above. Latter, a new group of bees was split into three groups and each group was trained using one of the three types of learning mentioned. The materials used in each of these types of learning was the same, a rounded platform and three balls on different distances from the platform's center. In a social learning setup, a pre-trained bee showed a trainee bee how to move the furthest ball to center of the platform. Similarly, in a *ghost* learning setup, a *ghost* demonstrator, namely a magnet operated by a researcher, showed the trainee bee how to move the furthest ball to the platform's

[32] All bees were pre-trained to find a reward (sucrose solution) when the ball was on the center of the platform.

[33] Also pre-trained in the same way as the group of experiment bees.

[34] In fact, all the bees from the experiment group were able to successfully carry out the task in all the trials, i.e. they got a 100% accuracy score, compared with a less than a 10% accuracy score from the control group. A paired t-test was computed to compare the accuracy means of both groups.

[35] In this pre-training phase, a square platform was used.

center. Finally, in a no-demonstration learning, a trainee bee was shown the ball already at the platform's center. In all the setups, a reward was given to all bees when the ball reached the platform's center. After training, a test phase followed where a bee could choose from any of the three balls in the platform to move to the center. Results show that bees trained using social learning achieved a much better accuracy score (around 99%) on the test phase than the other two groups. Similarly, bees trained using a *ghost* demonstrator had a better accuracy score (around 78%) than bees with no demonstrator (around 34%). Surprisingly, bees successfully solving the task, from the three groups, used much more frequently the closest ball to the center of the platform[36] despite the fact that during learning the demonstrator used the farthest ball. This result seems to show evidence towards a cognitive flexibility in bees by not copying a previous solution from a demonstrator, but rather learning a more abstract conceptualization of how to solve the task.

In a final test phase, the same bees were given the same experimental setup as described above, but the ball that was closest to the platform's center was from a different color. Bees chose the closest ball to the center, indicating that they were not paying so much attention to this characteristic, but rather conceptualizing the task in a more functional way. Overall, these results seem to show that, indeed, bumblebees have cognitive and behavioral flexibility in order to learn artificial tasks. These experimental setups are not likely to be found in the normal environment of bees. Therefore, bees could not have learned this previously by genetic predisposition or from other bees outside the laboratory. This throws light on the complex cognitive and behavioral capacities of bees and motivates to do further studies.

The two works we just described above proved psychological capacities of bumblebees and monkeys. Given that animals are not able to verbalize, these two works used behavioral analysis as a tool to correlate behavior with a cognitive capacity. As we saw, both works used controlled environments in order to guarantee an internal validity. Moreover, statistical tests were used in order to guarantee a correlation between independent and dependent variables, or to compare a property

[36]Chi-square statistics were computed in order to correlate the use of the closest ball with the success rate of the bees.

from two groups.

If we see behavioral analysis from the angle of researchers advocating for external validity in an experimental setup, we find that controlling for all possible confounding factors is not a plausible endeavour given the environment where the subjects are analyzed, namely the environment where the subjects operate[37] (or an environment similar to the operative environment), which is exactly the aim of these researchers, to understand the subject's behavior *in situ* (Dawkins, 2003; Epling and Pierce, 1986). Goals in particular for studying animal's behavior in such a type of environment is mainly related to welfare matters(Mench, 1998). Farm and zoo animals have a substantial change with respect to their natural habitats, which may greatly impact on their psychological and physiological systems. In the following paragraphs, we describe two works where farm and zoo animals are studied in their operating environments.

Birkett and Newton-Fisher (2011) studied the behavior of zoo-living chimpanzees in order to find any possible abnormal behaviors (abnormal when compared to the behaviors of wild chimpanzees living in their natural environment.) To do so, Birkett and Newton-Fisher (2011) observed the behavior of 40 chimpanzees distributed across 6 zoos. Since this type of study is done *in situ*, it is more similar to an observational study since controlling for confounding factors is very difficult.[38] However, Birkett and Newton-Fisher (2011) statistically studied the effect of possible confounding factors such as age, prior housing,[39] rearing history,[40] and sex. Furthermore, the behavior of the zoo-living chimpanzees was compared to that of wild chimpanzees. A list of abnormal behaviors was first collected; this list included 37 behaviors, such as *bite self*, *drink urine*, *pluck hair*, or *poke eye*, among others. Also, four measures of abnormal behavior were proposed in order to better characterize the chimpanzees' level of abnormality: Prevalence (proportion of chimpanzees, per group, displaying a certain behavior), frequency (how many times

[37] Lives in or works at.

[38] To control for any possible confounding factor, the experimenter would need to disrupt the environment of the subject under study, which by itself would be a confounding factor.

[39] For example, chimpanzees coming from a laboratory, from the wild, from another zoo, or from the entertainment industry.

[40] Such as mother-reared, hand-reared, and wild-born.

a behavior was displayed), duration (for how long a behavior was displayed), and diversity (number of behaviors per chimpanzee).

Birkett and Newton-Fisher (2011) found that all the chimpanzees under study displayed at least two abnormal behaviors, where the most prevalent behavior was *eat faeces*. They also discovered five more abnormal behaviors across all groups, namely *rock*,[41] *groom stereo-typically*,[42] *pat genitals*,[43] *regurgitate*,[44] and *fumble nipple*. In average, the number of abnormal behaviors displayed per group was $n = 18$ and no correlation was found with the number of chimpanzees integrating each group. In a per-group comparison, Birkett and Newton-Fisher (2011) found no significant difference of diversity and frequency (as defined above) in the abnormal behaviors displayed.[45] Similarly, Birkett and Newton-Fisher (2011) found no difference in the total duration of display of behaviors across groups.[46] In a per-chimpanzee analysis, however, Birkett and Newton-Fisher (2011) found significant differences in the frequency and duration of display of abnormal behaviors across groups: From 30 hours of observation, the median number of displayed behaviors was 1.45 behaviors per hour in a range of [0.13/hour-13.5/hour], while the median duration of an abnormal behavior was 1.32 hours in a range of [0.03-18.7].

As for possible confounding factors, Birkett and Newton-Fisher (2011) found no effect of sex, age, prior housing, and rearing history on frequency, diversity, and duration of the abnormal behaviors displayed by the chimpanzees across groups.[47] Furthermore, when the behaviors of the zoo-living chimpanzees were compare with those of wild-living chimpanzees, it was found that 17 out of the 37 abnormal behaviors displayed by the zoo-living chimpanzees were by far much more often than

[41] Sway from side to side either the whole body or just the head.

[42] Self-groom repetitively without any apparent goal.

[43] Self-touch genitals followed by licking hand.

[44] Voluntarily regurgitate usually followed by ingesting the vomit.

[45] Kruskal-Wallis H tests were computed for both measures, namely diversity and frequency, in order to compare differences across groups.

[46] Kruskal-Wallis H test was computed.

[47] Mann-Whitney U tests were computed to compare two samples and Kruskal-Wallis H tests were computed to compare more than two samples. For example, in order to compare if there is a difference between the samples of chimpanzees when divided by sex (male or female), a Mann-Whitney U test suffices to do so, while comparing if the samples of chimpanzees divided by rearing history (wild-born, mother-reared, or hand-reared) come from the same population then a Kruskal-Wallis H test suffices to do so.

from those conspecifics living in the wild.

Finally, these results show evidence that zoo-living chimpanzees are affected (presumably in a psychological way) when captive in an environment significant different from their natural habitat. Despite the fact that all chimpanzees under study were engaged in a social group, all of them displayed at least two abnormal behaviors. These abnormalities could not be explained by other factors, such as age, sex, rearing history or previous housing. Overall, it seems that the welfare of zoo-living chimpanzees is compromised despite the amenities provided by the zoos.

In a similar line or work as that of Birkett and Newton-Fisher (2011), Regan et al. (2014) studied the behavior of working donkeys with the aim of better characterize existing donkey ethograms. As Regan et al. (2014) noted, there exist equid ethograms, but these focus more on horse behavior than on donkey behavior. The importance of thoroughly characterizing in a systematic and ordered way donkey behaviors is to better understand, via behavior observation, when donkeys show pain, tiredness, distress, or discomfort, specially donkeys living or operating in environments other than their natural habitats. To work towards this goal, Regan et al. (2014) conducted an observational study of behavior of 21 working donkeys (12 females and 9 males); these donkeys were taken to a pen where they were given an 18 hrs. acclimatization period in order to get rid of possible confounding factors, such as tiredness, stress, fear, etc. After that, an observer recorded two types of behavior, namely postural behaviors (*ear position*, *ear level*, *head carriage*, and so on) and event behaviors (*walking*, *pawing*, *eating*, etc.); these observations were taken for two days.[48] In order to better characterize the behaviors observed, Regan et al. (2014) studied the effects of factors such as sex, time of the day when the behavior was observed, and period of observation across the two days (i.e. day 1 and day 2). Furthermore, an avoidance test was done where the observer approached each donkey and scored the level of proximity she was able to reach.

Regan et al. (2014) found that the behavior of *standing* was observed, in the

[48] The observations were recorded in the form of a count in the behavior space; that is, if a behavior A was observed for 10 minutes, then a variable indicating the number of times behavior A was observed was incremented by 1.

median, 78.4% of the time. Behaviors such as *walk* and *lie* were observed, in the median, 5.7% and 12.3% of the time, respectively. On the other hand, *rolling*, *self-grooming*, and *stretching* were rarely observed behaviors. Regan et al. (2014) discovered that sex played a crucial role in some of the behaviors observed.[49] Males usually positioned their head in a higher level than females, probably indicating a vigilant posture. On the other hand, males were more prone to bite than females, and females shacked their heads much more frequently than males. Behaviors unaffected by sex were *standing*, *walking*, *lying*, among others. Regan et al. (2014) also found that the time of the day influenced the donkeys to perform certain behaviors.[50] For example, some positions of both tail and ears differed by the time of the day; also, *lying*, *rolling*, and *head shacking* had different patterns across time. As for behaviors differing across periods of observation, only some positions of the ears changed from day 1 to day 2.[51] Finally, Regan et al. (2014) found that proximity levels of the observer towards the donkeys significantly differed by sex.[52]

These results throw light on how working donkey's behavior may differ according to sex, time of the day, and from day to day. Taking into account this variations may help clinicians to better assess behavioral patterns related to fatigue, pain, stress, and other illnesses. Moreover, this study has opened up the way for more research to be done on donkeys; the results obtained in this study may not generalize to donkeys working in different environments, or to donkeys who do not work.

3.2.2 Robustness

Robustness can be seen as the generalization ability of a machine learning system in difficult or challenging scenarios not commonly found at neither training nor test time. It has been tackled from different angles in the literature. In an adversarial setting, the objective is to fool a classifier, i.e. to make a system wrongly classify

[49] Mann-Whitney U tests were computed in order to find a difference in the two samples, namely the sample of observation from males and that from females.

[50] Friedman with chi-square tests were computed in order to find any correlation between the time of the day and the observed behaviors.

[51] Wilcoxon Signed Rank tests were computed in order to compare if the medians of the data from day 1 differ from the medians of the data from day 2.

[52] A Mann-Whitney U test was computed.

an instance; this is a way of exposing a weakness or a backdoor of a system. An example of an adversarial instance for an image classifier is the picture of an object, like a car, that had changed specific pixels so that the classifier is mislead into classifying it as other object, like a computer. These changes are so fine-tuned and small that for a human they are imperceptible yet for a classifier they are substantial. In the case of a parser (an NLP system), it would be desirable that this system is robust against a slightly wrong order of the words in a sentence that would remain understandable and close to a grammatical form for a human.

A second angle to probe a system is by measuring how well it avoids biases from the training data. For example, if an image classifier is trained with images where certain objects, such as a car or a washing machine, appear most frequently with either a man or a woman, it is desirable that given new pictures the system does not learn to predict the presence of a man just because a car appears there, and similarly in the case of predicting the presence of a woman given the presence of a washing machine.

A third angle has rather investigated how well NLP systems have captured certain linguistic phenomena relevant for the task they were trained for; therefore, in this type of work, an evaluation of the ability of a system to correctly handle difficult test instances is provided. In the following sections, we describe the three approaches in more detail.

3.2.2.1 Adversarial Instances

An adversarial instance is one that fools a classifier; the system assigns a wrong label to the instance by *confusing* the features of the instance to be from a different class. Previous works in the computer vision and machine learning communities have fooled representation learning systems (and other types of classifiers) to show weaknesses, for example, when recognizing images[53] (Nguyen et al., 2015; Szegedy et al., 2014), or when populating a knowledge base via transitive rules

[53]For example, a system may wrongly classify the image of a car as that of a panda; this can be done by adding some noise to the instance. However, the noise is imperceptible by a human, whom would correctly classify the image as being in the category *car*. Another way to fool a system is in the reverse direction, by transforming noise until the classifier recognizes it as the image of an object, but still being noise for a human.

(Minervini et al., 2017). The use of adversarial instances thus helps both to expose problems in robustness in systems and to formulate hypothesis of the reasons of these failures, as the hypothesis of Goodfellow et al. (2015), from the ML community, who attribute a lack of robustness to the linear nature of the systems. Other works have provided theoretical analysis for the upper bounds on the robustness of linear and quadratic classifiers (Fawzi et al., 2017); also, previous works have proposed formal measures in order to quantify robustness (Bastani et al., 2016), and has shown that there is room for further investigation (Carlini and Wagner, 2017).

However, most of the works have been done on computer vision systems and few work has been devoted to NLP systems. A relevant work in the NLP domain is the work of Jia and Liang (2017) where the vulnerability of reading comprehension systems is exposed. A reading comprehension system is a ReLe NLP system that answers a question regarding a specific paragraph of text. Jia and Liang (2017) added spurious sentences to paragraphs that did not change their semantics, but made the systems to output an incorrect answer. An interesting finding is that the systems have a hard time recognizing a spurious sentence as such. This behavior derives from the fact that words in the spurious sentence overlaps with the sentence containing the answer; i.e., the systems get *confused*. On the other hand, having an exact overlap of n words (n-gram) between a question and the sentence containing the answer allowed the systems to ignore the spurious sentence; also, having short questions helped the systems to be robust against adversarial instances. Overall, this work exposes a vulnerability in representation learning systems, and opens the door for further research on other NLP systems.

3.2.2.2 Bias Identification

Previous work has shown that representation learning systems are prone to capture biases from datasets, which may compromise generalization abilities by learning, for example, gender stereotypes in word embeddings (Bolukbasi et al., 2016) (from the ML community). From the NLP community, Zhao et al. (2017) showed that some ReLe systems capture spurious correlations among output variables, in a structured prediction task, based on the frequency of co-occurrence of the variables.

One example of this is wrongly predicting a woman in an image, instead of a man, based on the context surrounding the person, a kitchen. In a given dataset, it is more frequent to find pictures of women in the kitchen than those of men in the same context. Furthermore, Zhao et al. (2017) showed that ReLe systems amplify the bias found when predicting on test instances which may compromise the robustness of these systems.

3.2.2.3 Evaluation of NLP systems

Evaluating how well NLP systems capture specific phenomena of interest[54] is another way of testing the robustness and deficiencies of systems. This evaluation is complementary to the standard test set evaluation, which only provides an accuracy number that by itself does not inform what phenomena a system is good at and what types of errors it has made (Kummerfeld et al., 2012).

Previous work in the NLP community has investigated the robustness of both parsing (Bender et al., 2011; Rimell et al., 2009; B. Hashemi and Hwa, 2016) and machine translation (Isabelle et al., 2017) systems (including ReLe systems) where the common methodology is to focus on specific linguistic phenomena required for the task and then creating a test set with instances of such phenomena. This evaluation leads to a fine-grained analysis of how well the systems can deal with important linguistic constructions and what type of errors are found. In turn, this analysis can lead to the improvement of the systems, or the datasets, by identifying shortcomings. Common discoveries across the literature are, first, a new perspective of the system's abilities to perform the task at hand: The systems usually perform poorly in at least one phenomena; this is a perspective that arises only with the help of a dedicated evaluation and analyses. The second discovery, and a consequence of the first one, is the need for more evaluations and analysis that can help to disentangle the system's capabilities at different phenomena to better understand them.

[54] We make a difference to other line of work, as in (Linzen et al., 2016; Kuncoro et al., 2017) that is close to measuring robustness, but it may have a different objective. In this line of work, the research question is whether a ReLe system can learn certain linguistic phenomena, rather than asking how well it can capture the phenomena. This subtle difference leads to the creation of different types of datasets: In the first case, a dataset for training the system to learn the phenomena is built, while in the second case only a test set is built.

3.3 Internal Analysis and Extraction of Abilities Learned

In this section, we describe both how *internal analysis* in the neuroscience community aims to decode information from neuronal activity of people, and, in a similar vein, how experimental analysis done in the natural language processing community aims to extract *abilities learned*, such as hypernymy, from ReLe systems. Here, we will see how the analysis done to extract such abilities share many similarities with those studies from neuroscience. Even though we are not the first to propose the type of analysis done to ReLe systems to extract hypernymy, we contribute with drawing a parallel between this analysis and that from neuroscience. In Chapter 6, we will describe in a fine-grained detail how we are able to extract hypernymy from a ReLe system.

3.3.1 Internal Analysis

This type of analysis refers to studying the internal components of the subject under study. For example, studies in neuroscience answer research questions about what knowledge, memories, or abilities, humans have learned and are encoded in the neurons. More concretely, previous work in neuroscience has analyzed to what extent it is possible to decode information from brain readings from humans, i.e. to extract symbolic information from their neuronal activity. The basic methodology consists in obtaining readings from the brain via sensors, such as fMRI (functional magnetic resonance imaging), EEG (electroencephalogram), or MEG (magnetoencephalogram), which deliver a representation of neural activity.[55] This neural activity is read while the subject obtains a stimulus, such as an image (Schoenmakers et al., 2013; Sudre et al., 2012; Chan et al., 2011; Naselaris et al., 2009; Yargholi and Hossein-Zadeh, 2016), or a word (Sudre et al., 2012; Chan et al., 2011); for example, a person reads, or listens to, the word *dog* while sensors connected to the head read the activity from the brain. In this way, it is possible to obtain data where

[55] While fMRI provides a coloured representation of active brain areas due to the oxygenated blood passing by, EEG and MEG provide a representation of brain activity in terms of electric and magnetic fields respectively (Smith, 2013; Lopes da Silva, 2013).

the representation of the neural activity is associated with the stimulus. This data will then serve to train a decoder (usually a supervised classifier) that will extract patterns in order to predict at test time what is the stimulus a person is receiving just by looking at her neural activity. This methodology has been applied to decode different types of information besides words or images, such as movie clips (Nishimoto et al., 2011) and dreams (Horikawa and Kamitani, 2017).

One goal of decoding neural activity is to understand brain organization by identifying which parts of the brain encode which type of information, such as motor or cognitive information (Naselaris et al., 2011; Smith, 2013). More concretely, decoding neural activity –brain reading– is a method to better understand specific regions of the brain, previously unreachable or investigated using invasive mechanisms, via non-invasive mechanisms, such as fMRI. For example, Naselaris et al. (2011) propose some research questions answerable by decoding fMRI readings, such as investigating if certain types of semantic information can be found in specific regions of the brain, or if the information encoded in a specific region is correlated with some behavioral patterns. Naselaris et al. (2011) claim that it is possible to answer such questions with this methodology given that a) decoding models can be trained on different regions of the brain and it is just a matter of comparing decoding accuracy of the different decoding models to know which region encodes more information about certain task or stimulus, and b) a decoding model can be trained to predict behavioral patterns instead of images or words thus correlating fMRI readings with behavior. In the following paragraphs, we review some works in order to better exemplify and ground this area of research.

Chan et al. (2011) investigated whether it is possible to train a classifier to predict (decode) semantic information from EEG and MEG data; i.e., they investigated whether it is possible to decode the category of a stimulus presented to a subject (a person) from living and non-living categories.[56] Also, Chan et al. (2011) trained classifiers to decode a specific word that the subject was thinking of.[57] Chan et al.

[56]The stimulus was presented as both auditory and visual formats.

[57]To do this, Chan et al. (2011) asked the participant to press a button if the image of the object presented as stimulus was bigger than 1 foot; this allowed the participant to think of characteristics of such an object, which were recorded in EEG and MEG data.

(2011) trained one decoder, a support vector machine, for each subject under study for each of the two tasks from above.[58] Chan et al. (2011) acknowledged differences between brains and thus trained different classifiers for each person; however, part of the experiments was to investigate to what extent it is possible to train a *universal* classifier able to decode information for any of the participants.[59]

Results show that in both experiments decoders were able to extract meaningful information. When data from EEG and MEG readings were combined to train a living vs. non-living object classifier, accuracy scores went from 61% up to 91% and from 63% to 86% in auditory and visual modalities, respectively. On the other hand, accuracy scores for predicting a word ranged from 32% to 79% and from 66% to 97% in visual and auditory modalities, respectively, when using, again, a combination of EEG and MEG data. Furthermore, Chan et al. (2011) found that specific regions respond to specific types of information;[60] bilateral anterior temporal and inferior frontal regions seem to be correlated with information from non-living objects, while the left inferior temporal-occipital region seems to correlate with living objects. In the case of a decoder predicting words, the inferior occipital, inferior temporal, and bilateral anterior temporal regions seem to be more involved than other regions, i.e. they seem to contain more relevant information than other parts of the brain. Finally, Chan et al. (2011) showed that a *universal* classifier seems a plausible model for the participants of the study; accuracy scores for such a classifier were 30.2%, using visual information, and 41.3% when using auditory information; moreover, when compared to a random classifier with an accuracy of 20%, the trained classifiers seem to extract information that generalizes across participants.

Another example of brain decoding is the work of Sudre et al. (2012). Sudre et al. (2012) learned multiple output linear regression models as decoders in order to extract semantic features from a stimulus (a concept); for example, if the stimulus is the concept of *bird*, then a possible feature is *it flies*. The neural activity of

[58] One classifier was trained for distinguishing between living and non-living objects, and a multi-class classifier was trained for recognizing a specific word among a set of words.

[59] To answer this question, a classifier was trained using data from all subjects except for one, and tested on the data of the participant who was left out.

[60] This was done via an analysis of parameters of the classifiers learned.

nine subjects was recorded via MEG readings in order to learn the decoders. Each subject was presented with 60 concepts via a drawing and via a written word, and was asked some questions about that concept, for example *is it man-made?* in order to record neural activity of the subject when thinking about the semantic features. In this way, the input of the decoder was the MEG data recorded for a target concept and the output was a vector where each dimension contained an intensity value corresponding to each of the semantic features being contained in the target concept. In order to evaluate how accurate were the features extracted from the MEG data, Sudre et al. (2012) compared, for each participant, the output vectors containing extracted features of two different novel concepts[61] in order to see if each of the two predicted vectors corresponded with the true features of the two novel concepts; for example, given the novel concepts *bird* and *car*, in the form of MEG data, along with their true vectors of features v_1 and v_2,[62] the decoder would predict the vectors p_1 and p_2 and would compare if p_1 is closer to v_1 than to v_2 in the feature space (and similarly for p_2 with v_2 and v_1), which would mean that the decoder accurately predicted the semantic features corresponding to *bird* and *car*. Results show that it is possible to discriminate between two unseen concepts by extracting semantic features of each of them from MEG data, with an average accuracy of 91% across participants. Moreover, Sudre et al. (2012) found that the left inferior-parietal cortex and the left lateral occipital were relevant areas of the brain for decoding semantic information.

In a similar vein, Schoenmakers et al. (2013) proposed a decoding framework to reconstruct handwritten images of characters from fMRI data. Schoenmakers et al. (2013) implemented a probabilistic model, a linear Gaussian model, in order to decode the most likely image from neural activity, where a prediction is of the form $\hat{x} = arg\,max_x\{p(x|y)\}$, where x is a handwritten image and y is a fMRI reading. To do this, three participants were shown 60 instances of six handwritten characters (B, R, A, I, N, S) in order to record fMRI data. Schoenmakers et al. (2013) trained six

[61] These concepts were not part of the training data.

[62] For example, consider the vector v_1=[*it flies*, *it is an animal*, *it has wings*, ...] in the case of *bird*, and the vector v_2=[*it has wheels*, *it is man-made*, *it has windows*, ...] in the case of *car*.

different types of decoders per participant; one decoder for one character, another decoder for two characters, and so on until the last decoder which used data from the six characters. In order to evaluate the quality of the decoded images, statistical correlation measures were computed between the original and the decoded images. These correlations were compared against the correlations between the original images and randomly-constructed images. Results show that correlations between the original images and those decoded from fMRI data are significantly higher than those between the original images and the randomly-constructed images,[63] which means that the information encoded in the neural activity was relevant for decoding the images shown to the participants.

3.3.2 Abilities Learned

Previous works in the Natural Language Processing community have focused on extracting linguistic abilities learned by ReLe systems, such as hypernymy, meronymy, or other types of information, such as referential information. We devote this section to describe work on these phenomena.

3.3.2.1 Hypernym Extraction

The vector representations (embeddings) learned by ReLe systems are opaque and thus it is difficult to know what information they have captured. Previous work has tackled this problem by extracting certain linguistic phenomena from them, such as hypernymy.[64] There are two main methodologies for extracting hypernymy from word vectors, unsupervised and supervised, which we explain below.

Unsupervised Prediction of Hypernymy In an unsupervised setting, a directional measure of entailment[65] between two word embeddings is proposed (Lenci and Benotto, 2012; Santus et al., 2014a; Weeds et al., 2004; Rei and Briscoe, 2014). These measures are based on two types of features. On the one hand, the measures use features important for hypernymy that are assumed to be captured by the embeddings; for example, the concept *dog* shares features with the concept *animal*, such

[63] Average correlation coefficients for the decoded images were in the range of $\rho \in [0.4, 0.5)$.

[64] An example of two concepts holding in the hypernym relationship is *dog* and *animal*, since the former is a type of the latter.

[65] Hypernymy is a special case of entailment.

as *lives*, *eats*, and *procreates*. On the other hand, features that are not shared by the two concepts are also used because they can provide information of how different the concepts are and thus help to infer which concept is more abstract; for example, *barks* is specific to *dog* and is not a feature of *animal* which means that *dog* is a less abstract concept. In this way, a directional measure would consider such features to decide if the entailment relation holds in the vector space; i.e. given the vector representations of *animal* and *dog*, a classifier can predict that *animal* is a hypernym of *dog*. However, most of the directional measures were originally hand-crafted for distributional vectors and not for word embeddings. In a distributional vector it is clear whether a particular feature is captured from the data, since each dimension in the vector represents the presence of a feature. This intuition is not clear in word embeddings, since it is unknown if any of the features are captured in any of the dimensions.

Supervised Prediction of Hypernymy In a supervised setting, a classifier is trained to detect whether the hypernym relation between two concepts holds based solely on their word embeddings, i.e. the input to the learner are the representations of the concepts in embedding space. Therefore, the classifier is trained to predict that *animal* is a hypernym of *dog*. More concretely, the problem of extracting hypernymy from word embeddings is posed as a binary classification task. The input to a classifier is a transformed pair of word embeddings $T_k(\mathbf{w}_i, \mathbf{w}_j)$,[66] where each embedding corresponds to a concept; for example, w_i may represent the concept *dog* and w_j may represent the concept *animal*. The two most common transformations are vector difference $T_k(\mathbf{w}_i, \mathbf{w}_j) = \mathbf{w}_j - \mathbf{w}_i$, which we call *diff*, and vector concatenation, $T_k(\mathbf{w}_i, \mathbf{w}_j) = \mathbf{w}_i\mathbf{w}_j$, which we call *concat*. The output of the classifier is the probability that the concept represented by the embedding $\mathbf{w}_i$ is a hyponym of the concept represented by the embedding $\mathbf{w}_j$.

The experimental setting of this classification task can be viewed as a type of *laboratory* setting, since it takes place outside any downstream task, and a rigor-

[66] We use the notation w_i to refer to a concept and $\mathbf{w}_i$ to refer to the embedding representation of that concept. For example, *cat* is the concept to be represented and $\mathbf{w}_{cat}$ is the embedding of the concept.

ous control of the experimental conditions should be done in order to ensure that the decision of the classifier is only due to the word embeddings and not due to a confounding factor. We can see that the supervised approach to extract hypernymy from word embeddings is similar to the internal analysis from neuroscience. In this scenario, the analogous of a representation of neural activity (such as fMRI or MEG readings) in a ReLe system are its internal parameters, the word embeddings, and the stimulus are the two concepts to infer whether they hold in a hypernym relation (such as *animal* and *dog*). In other words, when studying a ReLe system, we try to decode (recover) semantic information, namely hypernymy, based on the internal representations of two concepts of the system under study; this is similar to work in neuroscience, where based on the fMRI or MEG readings recorded when a participant was presented with a stimulus, a decoder tries to recover either semantic features from the stimulus or the stimulus itself. To do this, in both cases, a decoder, usually in the form of a classifier, is trained to extract information from the neural representations of the participant's brain or from the the ReLe system's embeddings.

Previous works on the supervised approach have studied different ReLe systems and have proposed different types of classifiers; these previous works have also used different score measures and have produced different hypernymy datasets. As a consequence of combining different levels of these experimental variables, the experimental setting across the works has become heterogeneous to the point that the overall results are not clear, i.e. there does not seem to be a clear consensus of whether word embeddings capture hypernymy, despite the methodology being the same in all these works (Roller et al., 2014; Roller and Erk, 2016; Weeds et al., 2014; Vylomova et al., 2016; Levy et al., 2015; Fu et al., 2014; Necsulescu et al., 2015).

3.3.2.2 Other Abilities

Even though the literature has focused more on hypernymy, there are other abilities in the form of semantic and referential features that previous works have tried to decode as well. For example, Necsulescu et al. (2015) trained SVM classifiers

to decode meronymy, a semantic relation, from pairs of word embeddings; i.e., given the representations of two concepts, such as *door* and *house*, the classifier predicted if the former concept *is-part-of* the latter concept (in this example we see that a *door* is indeed a part of a *house*.) These classifiers were trained using data from WordNet, excluding concepts with multiple senses. Results greatly varied depending on both the corpus used for training the ReLe system containing the embeddings and the way the embeddings are combined when passed as input to the classifier; for example, when using the British National Corpus, F_1 scores ranged from 40% up to 58.7%, and when using the Wikipedia corpus, F_1 scores ranged from 59% up to 72.9%. These results show some evidence that word embeddings are able to capture this semantic relation.

Antonymy is another semantic relation considered in previous work. Santus et al. (2014b) investigated to what extent it is possible to extract antonymy from pairs of distributional vectors of words. To do so, Santus et al. (2014b) modified a metric from Information Retrieval, namely Average Precision, which evaluates the ranking abilities of systems when retrieving documents from a given query; Santus et al. (2014b) modified this metric in order to measure the relevancy of concepts that are contexts words[67] to the two target concepts; for example, the concept *food* is likely to be a common context word of both *dog* and *cat*. Santus et al. (2014b) took into account the ranking of these context words to obtain an estimate of how dissimilar are the two target concepts, where the ranks are computed based on the local mutual information of each context word to each target concept; these characteristics gave rise to the new metric called *APAnt*. Results show that *APAnt* worked better than baselines metrics: In a range of [0, 1], where 1 is the best score, *APAnt* obtained a score of 0.73, compared to scores of 0.56 and 0.55 of the two baselines, thus providing evidence of the relevance of the method and the possibility of decoding antonymy from distributed representations of concepts.

Another previous work on decoding information from embeddings is that of Gupta et al. (2015). In this work, Gupta et al. (2015) decoded referential informa-

[67] A context word is a word that appears to the left or to the right of the target word in sentences from a corpus.

tion from word embeddings of countries and cities (trained using word2vec); for example, population size, latitude and longitude coordinates, fertility rate, and so on. Gupta et al. (2015) used logistic regression models as decoders trained on data from Freebase. Since predicting the exact value for each referential attribute is very difficult due to the fine-grained value required, Gupta et al. (2015) evaluated the predictions as a ranking task. Information predicted for each referential attribute across countries was ranked from lower to higher and it was compared to the original rank; for example, if the countries in the ranked list of fertility rate are Spain, Greecy, UK, and Italy, and the prediction from the decoder ranked the countries in the same way, regardless of actual values predicted, then the evaluation of the decoder would be perfect. We note that the lists of countries (260 countries in total) and cities (1645 cities in total) were divided into training and test sets without overlap; however, when evaluating the ranked list of countries, all countries from training and test sets were taken into account. We also note that some attributes are binary, and thus an accuracy score suffices for evaluation. The results obtained show that the word embeddings encoded referential information; accuracy score of decoders of binary attributes of countries is 90% while that of cities is 99%. On the other hand, ranking scores are 0.22 and 0.25 (a ranking score of $r = 0$ is a perfect score) for countries and cities, respectively. These results seem to confirm the feasibility of decoding referential information, to some extent, from word embeddings.

Chapter 4

Representation-Level Analysis of *Model F*: Explaining Predictions

4.1 Introduction

Matrix factorization models (described in Chapter 2) have been widely deployed into systems and used in several tasks, such as recommender systems (Koren et al., 2009), information extraction (or knowledge base population) (Riedel et al., 2013), learning word embeddings (Levy and Goldberg, 2014; Pennington et al., 2014), and link prediction (Menon and Elkan, 2011).

There are several reasons for their wide popularity: a) scalability, since they are able to handle datasets with thousands of variables; b) good performance in terms of accuracy; c) fast training time; d) the lack of *vanishing gradients*, a problem present in other ReLe systems (see Section 2.1.1.2); e) the fact that they learn embeddings which can be used for transferring the knowledge learned to other tasks.

A popular matrix factorization system in the NLP community is *Model F* (Riedel et al., 2013). This system populates a cell in a matrix of relational data by predicting the likelihood of the truth value of a fact; examples of facts are $capitalOf(London, England)$ and $reviewMovie(Daniel\_Kahneman, Nobel)$. For such facts, *Model F* computes the likelihood of their truth value. In other words, *Model F* predicts the probability that the realization of a pair of entities e_i : $(Daniel\_Kahneman, Nobel)$ under a relation r_j : *reviewMovie* is true, i.e. $p(r_j(e_i) =$

$True$),[1] or more specifically $p(reviewMovie(Daniel\_Kahneman, Nobel) = True)$.

However, both the intricate internal mechanism and the complexity of *Model F* convert it into a black-box model. Thus understanding the *reasons* behind a particular prediction[2] y_i is difficult for a person. For example, suppose *Model F* makes the following prediction: $reviewMovie(Daniel\_Kahneman, Nobel) = True$.[3] The interpretation of this prediction states that Daniel Kahneman, an economics Nobel prize winner, reviewed a movie named the same as the prestigious prize; this is, indeed, an incorrect prediction. How can we know why the MF system made this prediction?

The most straightforward way to explain a prediction is to simply describe how the internal machinery of the system produced the prediction. The prediction is obtained by first computing the dot product of two embeddings, the embedding of the pair of entities $(Daniel\_Kahneman, Nobel)$ and the embedding of the relation *reviewMovie*; then, a sigmoid function is applied to the dot product, i.e. a confidence value is obtained based on the similarity of the two embeddings (see Equation 4.1); the more similar the two embeddings are, the more confident is the prediction. However, after this *mechanistic explanation*, do we know why the system predicted that Daniel Kahneman reviewed a movie named Nobel?

We argue that a mechanistic explanation,[4] such as the one above, is not the correct type of explanation to understand the *decision process* of *Model F*. First of all, the distributed representations (embeddings) have no clear direct interpretation, i.e. we do not know what information is encoded in each of the dimensions of the embeddings. Therefore, computing the similarity of the embeddings can only give us an intuition about a hidden statistical pattern captured from the data. What we need is an explanation meaningful for a human where the input of the system is related, in an understandable way, to the prediction.

---

[1] If this probability is thresholded then we call it a binary prediction where possible prediction values are *True* or *False*. We indiscriminately use y_i or $r_j(e_i)$ to refer to such a prediction.

[2] We consider a prediction to be the output of a system. We also refer to a prediction as the response of a system given that both are the same observable output variable.

[3] We consider this prediction to be binary after being thresholded at threshold α.

[4] See Section 3.1.2 for a comparison between a mechanistic explanation and a functional explanation, which is a type of explanation that goes in a similar vein to the explanation we propose for *Model F*.

Hence, we argue that the type of explanation needed is one that stems from an analysis at the representation and algorithm level[5] (see Sections 3.1.1 and 3.1.3). This type of explanation describes the decision process made by *Model F* by showing how the mapping from the input to the observed output is done (we elaborate on this point in Section 4.1.1.) This analysis can be captured by an interpretable proxy model.[6] Our objective is, therefore, to learn an interpretable proxy model that globally[7] captures the knowledge of *Model F* and it is able to explain any particular prediction y_i, in terms of a decision process, based on the inputs to the system.

At this point, however, one question arises due to the input-output structure of *Model F*, which is different from that of most of the black-box systems from the literature where a set of independent variables $\mathbf{x}$ is mapped to a label y (see Section 3.1.4 for a deeper description); for example, take a common black-box system, a neural network, that makes the same incorrect prediction as *Model F*, i.e. it predicts $reviewMovie(Daniel\_Kahneman, Nobel) = True$. The neural network would make this prediction based on known (and probably incorrect) facts about Daniel Kahneman and Nobel, such as $likes(Daniel\_Kahneman, Nobel) = True$ or $postulatedFor(Daniel\_Kahneman, Nobel) = True$. Nevertheless, *Model F* does not work in this way; it makes a prediction based only on the vector representations of both the relation (*reviewMovie*) and the pair of entities (*Daniel_Kahneman*, *Nobel*). Thus, while the explanation for the neural network would be based on an interpretable relation between the known facts and the prediction, the explanation for *Model F* would look like the mechanistic explanation described above. Then, we pose the following question: How can we obtain an input-output model from *Model F* in terms of relations? I.e., how can we relate a prediction $r_j(e_i)$ with other facts

[5]We set aside analyses at the computational and implementation level since sketches of both of them are provided in the original paper where *Model F* is introduced, i.e. in (Riedel et al., 2013), where the task carried out by *Model F* and its internal components are described. Thus, our goal is not to describe the task nor the internal mechanism of *Model F*, but rather to understand its decision process, i.e. to understand how a particular prediction is obtained.

[6]We will use the terms *proxy* and *descriptive* indiscriminately to refer to the concept of a model that acts as an interface to the user in order to understand a black-box system.

[7]This term refers to the scope of the explanation. As described in Section 3.1.4.2, a global explanation captures the whole knowledge encoded in the black-box system and thus it is able to explain any prediction.

$r_k(e_i)$ in order to build an interpretable explanation?

If we consider using a proxy model from the literature, then we need to conceptualize *Model F* as a multi-label classifier. Thus, in order to explain a prediction $r_j(e_i)$ in terms of other facts, we define the input space to be all relations $r_k \in R$ applied over the entity pair e_i, where R is the set of all relations. Similarly, we define the output space to be all relations $r_j \in R$ that are applied to any entity pair. However, this setting leads to a potential difficulty. There is now a significant difference in the size of the input domain of *Model F* with that of black-box classifiers from the literature; while in previous work the number of variables is usually in the hundreds or less, *Model F* is trained on around 4000 variables (relations $r \in R$). This fact impacts on the selection of the interpretable model due to scalability issues; for example, learning an interpretable model of arbitrary size, suitable to capture the knowledge of a multi-label classifier, such as logic rules, is NP-hard (Jacobsson, 2005). Then, another question arises. What interpretable model, able to capture a multi-label classifier, is feasible to learn in polynomial time?

In addition, we pinpoint another dimension to consider for choosing an adequate interpretable descriptive model, namely the expressiveness of the model. *Model F* is trained using a ranking loss function, where the objective is to rank true observed instances (facts) from the training set with a higher probability than false[8] ones. A natural question arising now is whether proxy models from the literature can capture the behavior of *Model F*.

Aggregating all the questions that have emerged so far, we pose the research question: What is a good interpretable proxy model for *Model F*?

As a final discussion on previous work, we consider the decompositional work of Yang et al. (2015) on learning logic rules from tensor factorization systems, a generalization of matrix factorization systems (Section 3.1.4.1.) This work is close to our objective but it is not clear to what extent we can apply their methodology for solving our problem. First, there are differences between black-box sys-

[8]False instances are actually unobserved facts. For example, the fact $cityIn(London, USA)$ does not exist in the matrix of training data $\mathbf{Y}$, but it is created to be used as a false observed instance, though it could be a wrong observation.

tems: The entities in *Model F* are not typed, thus we cannot distinguish if the entity *Daniel_Kahneman* is a person, or a company, or something else; Yang et al. (2015) use this knowledge as a heuristic to learn the logic rules, which ameliorates the complexity of the search of rules. Another difference is the way entities are represented; embeddings in *Model F* are learned for entity pairs, as opposed to having a single embedding for each entity. Finally, we consider the scalability and expressiveness of their proxy model: Precision scores of the rules extracted in (Yang et al., 2015) degrade considerably when the number of predictions is in the order of the thousands, which is the same order that *Model F* handles; such results seem to suggest that logic rules are not expressive enough to capture the knowledge from a matrix factorization system. Hence, given that it is not totally clear how to properly adapt the decompositional approach of Yang et al. (2015) to analyze *Model F*, we consider using a variant of their proxy model in a pedagogical approach.

Overall, there is no clear answer to what is a good proxy model for *Model F* that allow us to explain its predictions. Thus, in this chapter we seek to answer this question via a set of experiments. We investigate three interpretable models, two of them widely used in the literature, logic rules and decision trees, and a third one that we propose based on its structural and functional characteristics, a Bayesian network tree (Nielsen et al., 2008; Pacer et al., 2013; Koller and Friedman, 2009). We use the pedagogical approach to learn the proxy models. Then, we compare how faithfully they mimic the predictive behavior of *Model F* and how useful they are to explain particular predictions. Fidelity tests are carried out by comparing how similar the predictions of the proxy models are with those from *Model F* on unseen instances; i.e. we compare which proxy model's predictions better align to the predictions of the target system. In addition, we compare generalization abilities of the proxy models with respect to that of *Model F*, the more faithful a proxy model is to the black-box system the more similar the generalization behavior of the two systems.

As a final remark, we note that the main objective of this study is to evaluate what is the most suitable interpretable proxy model, among the three models

proposed above, for explaining predictions of *Model F*. We clarify that we operationalize the concept of interpretability as finding a proxy model that is able to faithfully mimic the predictive behavior of a black-box system. However, we do not aim to study how well a proxy model serves as an interface to a user in order to help her understand the black-box system; i.e., we refrain from evaluating our proxy models with a set of users. Even though we believe this type of evaluation is a requisite for deploying proxy models as applications into the real world, we leave this study as future work, due to the complexity imposed by such a study since this type of study would require methods from both psychology and human computer interaction disciplines, and it would probably need an entire chapter of its own (as the work of Huysmans et al. (2011)).

4.1.1 Interpretable Proxy Models As an Equivalent of Representation-Level Analysis

In this section, we tie the work of interpretability to work in cognitive science dedicated to the same purpose, namely to understand the inner working of an information-processing system via an input-output mapping process.

Previous work in machine learning has defined the concept of prediction explanation as explaining the relation between the response of a system and the input variables that elicited such a response. This relation between the *trigger* of a prediction and the prediction itself is represented by the structure and/or parameters of an interpretable descriptive model. This explanation suffices a logical connection between the system's response and the input variables. For example, the prediction $y_i = cityIn(London, England) = True$ can be explained based on the observation $capitalOf(London, England)$ and the rule[9] $capitalOf(a,b) \rightarrow cityIn(a,b)$, where the observed fact is part of the input domain and the rule used is part of the interpretable descriptive model.

We now argue for an interpretable descriptive model to embody an equivalent

[9]This rule does not necessarily embody a logic rule; it is rather a *generic* rule that can represent other types of structure, such as an association rule, an entailment in a Bayesian network, etc. The function of this rule is to describe a flow of information.

analysis of representation-level analysis as proposed by David Marr (Marr, 2010).[10] As described in Section 3.1, an analysis at the representation and algorithm level seeks to propose a suitable representation and an algorithm that explain how a system transforms an input to an output. On the other hand, a proxy model seeks to explain how a prediction was made. In both cases, the target is an explanation of how a decision is made by a system. Therefore, the objectives of an analysis at the representation-level and those from a descriptive model seem to be in concordance.

But, do these two type of analyses characterize the problem similarly and propose the same kind of explanation? We believe so. As we described in Section 3.1.1, an analysis at the representation and algorithm level aims to explain how a system transforms an input to its associated output; this explanation is not propose in terms of the physical components of the system, but rather it is proposed in terms of the sequence of steps that the system has to carry out; furthermore, the input and output of the system need to be represented in a suitable way for the proposed algorithm. On the other hand, an interpretable model aims to explain how a system makes a prediction; according to our definition of prediction explanation, we seek for a structure that associates input factors[11] to the response of the system. This structure is usually a machine learning model that is easy to understand for a person, i.e. a person can understand how the inputs are transformed to the observed outputs. Thus, an interpretable model is able to show the logic of how inputs are mapped to outputs, and we claim that this logic is suitable to be approached (approximated) as an algorithm, even though we may loose certain information. For example, in the case of a Bayesian network, if we are to fully describe the mapping process from input to output, then we are required to explain step by step the Bayesian network's probabilistic inference process (see Section 2.3.1); nevertheless, this inference procedure is very complex to understand for a user, and the size of the resulting explanation can be overwhelming. Thus, as we will see in our ex-

[10] We do not claim that by tying a proxy model with analysis from cognitive science it means that we intend to describe any *mental* or some sort of cognitive capacities. Even though some machine learning models may be viewed as a type of cognitive model (Griffiths et al., 2008), we step aside of any claim about such capacities.

[11] Observed variables from the input domain.

periments, we need to reduce the complexity of the explanation by simplifying the description of the Bayesian network's inference process to a simpler logic of how the inputs are transformed to the outputs, namely as a sequence of entailments.

In a more detailed comparison of an interpretable model with a representation-level analysis, the structure of an interpretable model serves to define the representation for both inputs and outputs; and when this structure is coupled with an inference process in order to obtain a prediction, i.e. to map an input to an output, then we are able to obtain an algorithm that explains the mapping process. For example, if we learn a set of logic rules as an interpretable model of a system, then we constrain the representation of inputs and outputs to be within this formalism; thus, inputs and outputs can be represented as logic predicates. In logic, a common inference algorithm, i.e. a process for obtaining a prediction, is *modus ponens* (see Section 2.4.1), which is applied over the set of logic rules learned. Based on this, we can obtain an algorithmic approximation that describes how we are able to map inputs to outputs.[12]

Furthermore, there are two other similarities between the two types of analysis. First, the explanations obtained in both cases are distinct from explanations that describe the physical mechanisms of the system under study. Second, as proposed by Marr (2010), a representation-level analysis can benefit from having some knowledge of the system (for example, whether it operates in parallel or in serial mode); in the same way, an interpretable model can benefit from knowing characteristics of the system, for example, in our case, knowing that *Model F* was trained using a ranking loss, can help us propose an interpretable model that is able to rank items, such as a Bayesian network, and moreover, we are able to propose a more suitable evaluation for our interpretable model.

Overall, we can see that both types of analysis lead to very similar types of explanations and both work under very similar schemes; furthermore, both analy-

[12]For example, the prediction $y_i = contributesToEconomy(London, England) = True$ can be explained by the observed fact $capitalOf(London, England)$, the intermediate factor $cityIn(London, England)$ and the rules $r_1 : capitalOf(a,b) \rightarrow cityIn(a,b)$, $r_2 : cityIn(a,b) \rightarrow contributesToEconomy(a,b)$. Thus, by describing the application of *modus ponens* over this set of rules and facts in a step-by-step format, we would obtain an explanation of the above prediction in an algorithmic format.

ses share some philosophical aspects, such as keeping away from fully describing physical components (though benefiting from knowing physical aspects of the target system) and instead restricting their scope to explain a mapping process. Therefore, we claim that an interpretable model is equivalent to a representation-level analysis.

4.2 Problem Definition

We aim to extract the knowledge encoded in the matrix factorization system *Model F* in the form of an interpretable descriptive model, for the purpose of prediction explanation. We define the problem as: Given the target system $Model\_F : \mathbf{X}_{m\times n} = \mathbf{U}_{m\times k}\mathbf{V'}_{k\times n}$, where a) the system has been learned in a transductive manner, b) the learned factors $\mathbf{U}$ and $\mathbf{V}$ are vector representations (embeddings) of two set of features: pairs of entities (for example $(London, England)$), and relations between entities (for example *capitalOf*), and c) a prediction $y_{ij} = sigmoid(U_i V'_j)$ has possible values $y_{ij} \in \{0,1\}$ after being thresholded. Then, our objective is to learn a *human-interpretable* descriptive model of the form $\mathbf{x} \rightarrow y$ that accounts for the reasons that elicit a particular prediction y_{ij}. In other words, we aim to model the associations among a set of input and output variables[13] that explain the decision process of how a prediction y_{ij} was obtained by the given matrix factorization system.

This problem can be decomposed into sub-problems. First, we need to treat *Model F* as a multi-label classifier instead of a transductive system. A transductive system[14] does not learn a function that maps from input to output, rather it works in an instance-based manner. *Model F* can only predict a label[15] for those cells c_{ij} where an embedding for the relation r_j and an embedding for the entity pair e_i have been learned. Thus, we need to find a way to treat *Model F* as a multi-label classifier in order to define an input and output space for the proxy models. And second, we need to explore interpretable proxy models that are scalable and expressive enough to capture the whole knowledge of *Model F* which has around 4000 relations $r_j \in R$ and has been trained using a ranking loss function.

[13] We consider input and output variables to be the relation types $r_j \in R$

[14] K-nearest neighbors is another type of transductive system. For another example, and some discussion, of transductive systems we refer to (Gammerman et al., 1998).

[15] The truth value of a fact.

4.3 Research Questions and Hypotheses

We consider as proxy models two previous models from the literature, logic rules (LR) and decision trees (DT), and we propose a new proxy model, namely a tree-structured Bayesian network (BN tree). This setting leads to the following questions and hypotheses.

Research Questions

1. How can we treat *Model F* as a multi-label classifier in order to define both input and output spaces for learning an interpretable proxy model?

2. How can we produce training data in order to train the proxy models to mimic the behavior of *Model F*?

3. Can we train logic rules, decision trees, and a Bayesian network using the same training regime?

4. How can we evaluate the fidelity of a proxy model? Can we use the metrics from the literature?

5. What is a good interpretable proxy model that is able to capture the whole knowledge of *Model F*?

    - What proxy model can faithfully mimic the predictive behavior of *Model F*?
    - What proxy model is both scalable and expressive enough to capture the knowledge of *Model F* while its training regime takes polynomial time?
    - Is any of the proposed proxy models an equivalent of *Model F* in terms of functional behavior?

6. How can we explain a particular prediction of *Model F* using a proxy model?

    - How does an explanation, from each of the proposed proxy models, look like?

Hypotheses

1. We hypothesize that a tree-structured Bayesian network (BN) and decision trees may be better able to capture the predictive behavior of *Model F* than logic rules, given their ability to handle probabilities and given the evidence in (Yang et al., 2015) for logic rules not fully capturing the behavior of tensor factorization systems.

2. We hypothesize that previous metrics to measure fidelity, namely F_1 and accuracy, may not work properly in our problem since *Model F* was trained using a ranking loss function; thus we may need to use an appropriate metric for measuring a ranking behavior, such as precision-recall curves.

4.4 Contributions

- Our main contribution is that we study a new type of black-box system, *Model F*, a transductive matrix factorization system trained as a ranking system, rather than as a classifier as most of the systems from previous work.

- Also, we propose a new type of interpretable proxy model, a Bayesian network tree, and we compare it with two widely used proxy models from the literature, logic rules and decision trees.

- In addition, we propose a new evaluation metric of fidelity, namely precision-recall curves, and we compare this metric with those from the literature, namely F_1 and accuracy.

- Furthermore, we connect this line of work to that from cognitive science – representation-level analysis– by describing how both share similar objectives and how an interpretable model leads to a very similar type of explanation. (see Section 4.1.1.)

4.5 Scope and Limitations

We only study one ReLe system, namely *Model F*. Thus, our objectives are focused on understanding this specific system. Analyzing any variant of *Model F*, or any

other matrix factorization system, is out of our scope. Hence, both our results and our findings are only descriptive of and pertinent to *Model F* and they may not generalize to any other system. We leave as an open question for future work to what extent our findings can be indicative of the behavior of other ReLe systems.

As we have described before, we aim to explain the behavior of *Model F* from a decision process perspective; i.e., we aim to understand how *Model F* makes a decision (a prediction) by describing a plausible way in which it maps an input to an output. However, we do not aim to explain how other factors contribute to the behavior of our system under study; i.e., we do not consider in our paradigm how the internal components of *Model F* give rise to the behavior observed, we neither consider how changes in the data may affect its behavior. These types of explanations correspond to different perspectives and different methodologies. Similar to work in cognitive science, we mainly focus on the representation and the input-output mapping process suitable for explaining the decision process of the system under study.

Furthermore, operations such as debugging the system or finding errors in the data are possible applications for which our work can be helpful, but we leave them as future work. We are concerned and interested in understanding *Model F*, and any possible engineering application derived form this understanding is welcomed as future research.

Another limitation of our work is that we do not prove whether *Model F* has learned any ability, such as a linguistic ability, a hierarchical structure, or phenomena of the sort. The proxy models that we obtain serve to a particular purpose, namely explaining the decision process of *Model F*; investigating to what extent these models can also serve to explore the abilities learned by the system under study is out of our scope and out of our research framework.

4.6 System Under Study

In this section we describe our target system, *Model F*, as well as the data used for training it.

4.6.1 Training Data

Model F was trained using two sources of data, the NYT corpus (Sandhaus, 2008), a collection of newspaper articles, and Freebase,[16] a structured database with information about the world in the form of facts. These two datasets were selected due to their high coverage of information. The NYT corpus spans throughout 10 years of news, and Freebase contains 1.9 billion of triples. Furthermore, as explained in (Riedel et al., 2013), by using data in the form of both natural language sentences and structured triples, a system is then able to learn and reason in both structured and unstructured modes.

In order to train and test *Model F*, a collection of articles were taken from the NYT corpus as training data, namely those published after the 2000; similarly, a collection of facts of the form $relation(entitiy_1, entity_2)$ were taken from Freebase (half of the facts from Freebase); an example of a fact is $capitalOf(London, England)$ where *capitalOf* is a relation applied to a pair of entities, namely $(London, England)$. In order to obtain facts of the same form from the NYT articles, surface patterns between *entity1* and *entity2* were extracted using a dependency parser; thus, a fact extracted from the NYT corpus looks in the following way $entitiy_1$ *surface_pattern* $entitiy_2$, for example *London_is_the_capital_of_England*. Each fact from the NYT corpus can be seen as $surface\_pattern(entitiy_1, entity_2)$. Finally, facts from both sources, the NYT corpus and Freebase, are joined together to form a matrix of relational data $\mathbf{X}$ which is the training data (see Figure 4.1 for an example of a relational matrix.) Each cell $x_{i,j} \in \mathbf{X}$ indicates if a fact has been observed on either of the two sources. For example, if $capitalOf(London, England)$ has been observed, then its corresponding cell is filled with 1, otherwise the cell is filled with the missing-value symbol: ?.

In summary, the matrix of relational data $\mathbf{X}$, which is used as training data for *Model F*, encodes facts extracted from both the NYT corpus and Freebase. Each row of the matrix represents a pair of entities $e_i : (entitiy_1, entity_2)$, for example $(London, England)$, and each column represents a relation r_j between entities, for

[16] The official website was www.freebase.com but it was shutdown.

example *capitalOf*. We indiscriminately use the notation r_j to refer to either a relation or a surface pattern.

4.6.2 Model F

Model F learns a k-dimensional vector representation –an embedding– for each entity pair $e_i \in \mathbf{E}$ and for each relation $r_j \in \mathbf{R}$. For example, the pair of entities $(London, England)$ and the relation *capitalOf* can be represented with the k-dimensional vectors $[4.34, ...2.45, 3.45]$ and $[4.32, ...1.34, 6.12]$, respectively. *Model F* then uses these embeddings to obtain a fully-populated matrix $\mathbf{Y}$; the objective for building this matrix is two fold. First, to recall the information of the matrix $\mathbf{X}$; second, to predict values for those missing cells in $\mathbf{X}$. In other words, the matrix $\mathbf{Y}$ is an approximation (a reconstruction) of the matrix $\mathbf{X}$.

Thus, in the matrix $\mathbf{Y}$, each cell $y_{i,j} \in [0,1]$ can be seen as the prediction of a fact. This prediction can be interpreted as the probability of obtaining a *True* value when a pair of entities e_i bounds a relation r_j, i.e. $p(r_j(e_i) = True)$. For example, suppose the prediction $p(cityOf(London, England))$;[17] this prediction states the confidence of *Model F* towards the fact that London is a city of England. Each prediction $y_{i,j}$ is obtained by applying the sigmoid function to the dot product of two embeddings; these two embeddings correspond to that of the entity pair e_i, and that of the relation r_j, denoted by $\mathbf{u}_i$ and $\mathbf{v}_j$ respectively. The mathematical form of a prediction is:

$$p(r_j(e_i)) = sigmoid(\mathbf{u_i}\mathbf{v_j'}) = \frac{1}{1+e^{-(\mathbf{u_i}\mathbf{v_j'})}} \tag{4.1}$$

To see an example of a matrix $\mathbf{Y}$ where all cells have been predicted (populated) see Figure 4.2. This matrix represents the reconstruction of the matrix of data shown in Figure 4.1. This reconstruction is obtained by applying Equation 4.1 to predict all cells $y_{i,j} \in \mathbf{Y}$.

In order to learn the matrix $\mathbf{Y}$, each row $\mathbf{X}_i$ from matrix $\mathbf{X}$ is taken as a training instance; i.e., each row is a vector of i.i.d.[18] random variables, and the factors $\mathbf{U}$

[17] We abandon the use of a truth value for simplification and clarity.
[18] Independent and identically distributed.

Matrix **X**	*liveIn*	*workIn*	*bornIn*	*diedIn*
(John, Spain)	1	?	1	?
(Lily, India)	?	1	1	?
(Carl, India)	?	1	?	?
(Sarah, USA)	1	?	?	1

Figure 4.1: Example of a matrix of relational data. 1 indicates an observed fact, ? indicates missing value (unobserved fact.)

Matrix **Y**	*liveIn*	*workIn*	*bornIn*	*diedIn*
(John, Spain)	0.99	0.95	0.99	0.70
(Lily, India)	0.78	0.98	0.97	0.49
(Carl, India)	0.80	0.97	0.90	0.35
(Sarah, USA)	0.99	0.95	0.90	0.98

Figure 4.2: Example of reconstruction of a matrix of relational data where all cells are populated. Each cell indicates the probability of a predicted fact to be true.

and **V**, the embeddings of all entity pairs and relations, are the parameters to be estimated. A canonical loss function to minimize, in order to approximate the matrix **X**, is the sum squared error: $\sum_{i,j}(x_{ij}-y_{ij})^2$, which is equivalent to minimizing the Frobenius distance between **X** and **Y**, $|\mathbf{X}-\mathbf{Y}|^2_{Frob}$. However, the matrix of data does not have any negative instances to learn from, i.e. there are not any facts known to be *False*; only positive instances are provided. Therefore, minimizing the squared distance shown before may lead to overfitting. Then, the optimization function used to train *model F* is a logistic loss function:

$$argmax_{\mathbf{U},\mathbf{V}} \sum_{x^+_{ij}\in O} \sum_{x^-_{ik}\notin O} log(sigmoid(\mathbf{U}_i\mathbf{V}'_j - \mathbf{U}_i\mathbf{V}'_k)) \tag{4.2}$$

Where x^+_{ij} and x^-_{ik} indicate observed and unobserved instances (cells), respectively. The objective is to obtain parameters that strengthen the inequality $p(x^+_{ij}) > p(x^-_{ik})$; i.e., observed cells receive a higher score than unobserved ones. In order to minimize the loss function in Equation 4.2, stochastic gradient descent is applied.

4.7 Methods and Materials

We learn three types of interpretable proxy models from the matrix factorization system *Model F* (Riedel et al., 2013): Logic rules (LR), decision trees (DT), and a Bayesian network tree (BN tree). As explained in Section 4.1, our objective is to learn a proxy model that 1) captures all the knowledge of *Model F* and 2) explains particular predictions. In order to guarantee fidelity of the proxy model to the predictive behavior of *Model F*, we propose precision/recall curves as a scoring metric of faithfulness.

In this section we explain the setting for learning the descriptive models. We provide both the methods used for inducing each of the descriptive models and a description of the data used for learning these models.

4.7.1 Data

In order to learn the descriptive models, we use the predictions from *Model F* as training data. We use the embeddings learned by *Model F* to construct the matrix of predictions $\mathbf{Y}$ where each row corresponds to an entity pair and each column corresponds to a relation type (see an example of such a matrix in Fig. 4.2). Each prediction is thresholded at $\alpha \in [0,1]$, i.e. we obtain training sets from the cell-wise thresholded predictions. But, since the training data for learning a Bayesian network or a set of logic rules (unlabeled data) is in a different regime than the training data required for learning a decision tree (labeled data), we need to create two different training datasets, namely D_U and D_S; therefore, we need to treat *Model F* as both a joint model and a local classifier in order to obtain the training data for all the proxy models. En either case, we define 19 Freebase relations as target variables for test evaluation (see Figure 4.3 for the complete list of target variables.)

On the one hand, we need a set of unlabeled data for learning both logic rules and a Bayesian network. A Bayesian network models a joint probability distribution; thus we need a training set where an instance $\mathbf{x}_i$ is the joint observation of a set of random variables $x_1, x_2, ..., x_n$. This training data can also be used for learning logic rules: We can frame the task of learning rules from *Model F* as that of the *market basket analysis* (Agrawal et al., 1993). In this task, a matrix of transactions

Target variables
business/person/company
location/location/containedby
book/author/works_written
people/person/nationality
organization/parent/child
people/deceased_person/place_of_death
people/person/place_of_birth
location/neighborhood/neighborhood_of
people/person/parents
business/company/founders
film/film/directed_by
sports/sports_team/league
sports/sports_team/arena_stadium
sports/sports_team_owner/teams_owned
broadcast/broadcast/area_served
architecture/structure/architect
music/composer/compositions
people/person/religion
film/film/produced_by

Figure 4.3: Freebase relations used as target variables to test the proxy models.

is used for discovering rules, where rows corresponds to transactions and columns to products. Thus, we take the columns of the matrix populated by *Model F* (variables $x_1, x_2, ..., x_n$) as either products (as in the market basket analysis task) in order to learn logic rules, or as random variables in order to learn conditional probability distributions for a Bayesian network. On the other hand, decision trees are a discriminative classifier, so we need a supervised training set where each input vector $\mathbf{x}_i$ is labeled by *Model F*. For this purpose, we use input vectors from the training data used for learning *Model F* (described in Section 4.6.1); the label y_i for each vector comes from the response of *Model F*.

We show how an unsupervised and a supervised instance looks like with the following examples. In Example 4.3 we have a training instance, $\mathbf{x}$, filled with several facts. A fact $r_j(e_i)$ is the realization of a relation under an entity pair, such as $capitalOf(London, England)$. Each fact is a prediction of *Model F*. Then, we

take each relation as an independent variable, and each fact as a prediction. Since this is an example of an unsupervised instance, we do not need to label it. Then, in order to obtain more instances, we apply the relations to different entity pairs. In this case, we build an instance as described above because we need to model *Model F* as a multi-label classifier, where the output space is all the relations $r_j \in R$.

On the other hand, in Example 4.4 the training instance is of the form $(\mathbf{x}, y)$ where, again, $\mathbf{x}$ corresponds to the independent variables and y is a label. Nevertheless, the input vector is not filled with predictions from *Model F*, unlike the unsupervised instance in Example 4.3; it is rather filled with observations from the training data used to learn *Model F*. Then, the label of $\mathbf{x}$, namely y, is a prediction of *Model F*. Thus, the prediction of $containedBy(London, England)$ is the only prediction of *Model F*. The point for building a supervised instance in such a way is because we are interested in *Model F* as a binary classifier, where the output space is a single relation r_j.

(4.3) $\mathbf{x}$: $[capitalOf(a,b)$, $cityIn(a,b)$, $worksIn(a,b)$, $livesIn(a,b)$, $containedby(a,b)]$
Where a : *London* and b : *England*

(4.4) $(\mathbf{x}, y)$: $([capitalOf(a,b)$, $cityIn(a,b)$, $worksIn(a,b)$, $livesIn(a,b)]$, $containedby(a,b))$
Where a : *London* and b : *England*

More formally, we construct two different training sets: D_U and D_S. The first dataset is used for learning logic rules and a Bayesian network tree. In D_U, a vector of attributes $\mathbf{x}_i = (x_1, x_2, ..., x_n)$ corresponds to the i-th row of the matrix of predictions of *Model F*; i.e., a training instance is the set of predicted facts for a single pair of entities e_i across all relation types from r_1 to r_n. For example, suppose again the entity pair $(London, England)$ and the relations r_1 : *capitalOf*, r_2 : *cityIn*, r_3 : *worksIn*, and r_4 : *livesIn*; suppose the thresholded predictions of *Model F* for each realization are as follows: $[1,1,0,0]$. Then, this vector of predictions become a training instance $\mathbf{x}_i$.[19] In the case of D_S, a training instance is of the form $(\mathbf{x}_i, y_i)$,

[19]The same training instance has different interpretations depending on the model using it. For

Dataset	Form of Instances	Independent Variables	Dependent Variables	Test variables
D_U	$\mathbf{x}$	columns of *Model F*	none	19 vars. in Fig. 4.3
D_S	$(\mathbf{x}, y)$	columns of NYT corpus	19 vars. in Fig. 4.3	19 vars. in Fig. 4.3

Table 4.1: Description of the training datasets used for learning the descriptive models.

where the input vector $\mathbf{x}_i$ comes from the i-th row of the original training sets used for learning *Model F*, namely the NYT corpus and Freebase. And the label y_i corresponds to *Model F*'s prediction of 19 Freebase target variables[20] shown in Figure 4.3. The resulting datasets, D_U and D_S, consist of 4111 and 5007 input variables, respectively. While instances of D_U have no class label, each instance of D_S contains 19 class labels, the Freebase variables from Figure 4.3; alternatively, we can see training set D_S as a set of 19 training sets. (This is actually how we will use dataset D_S, we will split it into 19 training sets and we will learn one decision tree for each.) Both datasets contain 39864 training instances. See Table 4.1 for a summary of both datasets.

The test data to evaluate the proxy models is drawn from a portion of the predictions of *Model F*; these cells are not used at training time for training neither *Model F* nor the proxy models. We use the Freebase variables shown in Figure 4.3 as test data. Each variable corresponds to a column from *Model F*. As we described above, some instances are used at training time in order to learn both their embeddings and the proxy models; however, some instances are not used for training and thus are used as test data.

4.7.2 Choice of Proxy Models

Some of the aspects to take into consideration for choosing a descriptive model are the interpretability (how understandable is the model for people) (Huysmans et al., 2011; Freitas, 2014), the fidelity (to what extent it can capture the knowledge of the target black-box system), and the easiness of training. Another fourth aspect some-

learning a BN, the training instance is seen as a set of realized random variables; when learning logic rules, the same instance is taken as a set of realized predicates.

[20]We note that these 19 variables are included in the training instances of D_U as independent variables rather than as class labels.

times not taken into account is the accuracy of the descriptive model to generalize to unseen instances; this aspect is often ignored because, usually, the objective of a proxy model is to explain predictions of a black-box system and not to substitute it. These aspects lead to two trade-offs that we take into account for selecting a proxy model: Fidelity vs. interpretability and fidelity vs. generalization.

Fidelity vs. Interpretability Usually, the more faithful is the proxy model to the black-box system, the less interpretable it is for people. A proxy model that perfectly mimics the target black-box is the black-box itself; in this scenario, we loose all interpretability to gain complete fidelity. On the opposite side, an interpretable model, such as logic rules or linear regression, will give us near-optimal interpretability at the cost of fidelity (specially for state-of-the-art representation learning systems.) Thus finding a descriptive model that is able to mimic the decision process of the black-box system while remaining simple for a person to understand such process is one of the main objectives in the task of understanding the black-box system.

We hypothesize that a tree-structured Bayesian network may represent a good fidelity-interpretability trade-off. Bayesian networks are well known classifiers that have achieved state-of-the-art performance (Friedman et al., 1997; Janssens et al., 2004), and have been used in several domains, including ranking (Chapelle and Zhang, 2009), an ability learned by *Model F* at training time. This success of BNs is due to both their interpretable characteristics and their ability to capture joint probabilities and compute Bayesian probabilities (Pacer et al., 2013; Griffiths et al., 2008). However, due to the complexity in learning them from data we restricted ourselves to a simpler structure, namely a tree structure, though this choice may imply a loss in fidelity. Decision trees, on the other hand, are able to compute conditional probability estimates, while providing interpretable explanations. But, we are not sure to what extent they may capture the ranking behavior of *Model F*, as previous work has shown their poor ability in ranking problems (Provost and Domingos, 2003). We hypothesize that DTs may represent a fair trade-off, but not as good as that of the Bayesian network. Finally, we hypothesize that logic rules

may not represent a good trade-off. Logic rules are easy to understand, but they are not probabilistic models; thus, they may not be able to fully capture the behavior of *Model F*. It seems, then, that despite being an understandable model (and widely used in the literature), logic rules are less likely to faithfully mimic *Model F*'s behavior. (See Sections 2.3.1, 2.2.3, and 2.4.1 for background information on each of these models.)

Fidelity vs. Generalization Fidelity of a descriptive model to a black-box system is at odds with generalization (Zhou, 2004); i.e., if the descriptive model is able to faithfully mimic the predictive behavior of the target system then it is expected to make the same mistakes and have the same hits. Generalization ability of the proxy model to test instances is thus restricted by the knowledge learned from the black-box system. Then, if our objective is to have a proxy model that replaces the black-box system we should consider this trade-off. Since our objective is rather to explain predictions from *Model F*, we do not investigate in depth this trade-off, though we compare the generalization ability of our proposed proxy models to show their performance on unseen test instances.

4.7.3 Learning Proxy Models

4.7.3.1 Learning Logic Rules

We learn first-order Horn clauses of the form $\forall x_1 \forall x_2 A(x_1, x_2) \rightarrow B(x_1, x_2)$. This type of definite clause can be seen as a universally quantified implication. Given that learning logic rules of arbitrary size is NP-hard, we restrict the size of the rules to be $|R| = 1$ (one predicate in the body) in order to low-down the complexity to $O(n^2)$, where n is the number of predicates. Each rule is range restricted, i.e., the arguments x_1 and x_2 in the body predicate A must be the same as those in the head predicate B. An example of a logic rule is $\forall x_1 \forall x_2\, capitalOf(x_1, x_2) \rightarrow cityIn(x_1, x_2)$.[21]

We extract logic rules from *Model F* using its predictions as training data (dataset D_U). We apply a variant of the *A priori* algorithm (Agrawal and Srikant, 1994) used for the market basket analysis task. We compute the mutual information

[21] We recall that predicates A and B correspond to relations (columns) r_j, r_k from the matrix reconstructed by *Model F*.

(Equation 4.5) between each pair of predicates A and B as a support measure for accepting a rule. If the statistical dependence between A and B is greater or equal than a fixed (manually set) threshold then we accept a symmetric implication rule $A \leftrightarrow B$. In order to determine the direction of this rule, we compute a confidence value by comparing conditional probabilities: $if\, p(A|B) > p(B|A)\, then\, B \rightarrow A, else\, A \rightarrow B$. The algorithm is shown in Algorithm 2.

$$MI(A,B) = \sum_{a \in A, b \in B} p(A,B) log \frac{p(A,B)}{p(A)p(B)} \tag{4.5}$$

A, B: variables corresponding to predicates
D: training data
learnLogicRules(D){
foreach A *in* D **do**
  foreach B *in* D *where* $B \neq A$ **do**
    MI_{AB}=mutual information[A, B]
    if $MI_{AB} \geq \tau$ **then**
      if $p(A|B) > p(B|A)$ **then**
        $\forall x_1 \forall x_2 B(x_1,x_2) \rightarrow A(x_1,x_2)$
      else
        $\forall x_1 \forall x_2 A(x_1,x_2) \rightarrow B(x_1,x_2)$
      end
    else
      not accept rule
    end
  end
end
}

Algorithm 2: Algorithm for learning a set of logic rules using a variant of the *A priori* algorithm. Threshold τ indicates the criterion for accepting an implication rule between two predicates.

We choose mutual information as a support measure because it assigns monotonically increasing values to more statistically dependent variables. However, one problem with this measure is its variability with null entries, i.e. the amount of null entries in the variables A and B may affect the mutual information between the two. On the other hand, an advantage of mutual information is its suitability for discovering classification rules since it considers the joint distribution of the antecedent and the consequent of each rule. In previous experiments we tried to use an induc-

tive logic programming method as a baseline, but due to both scalability issues and the requirement of negative examples (also noted in (Galárraga et al., 2013)) we refrained from using it.

4.7.3.2 Learning Decision Trees

We learn CART decision trees using the software *Rpart* (Therneau et al., 2014). We learn one decision tree for each of the target variables in Figure 4.3. We use the dataset D_S to do so. As explained in Section 4.7.1, the input variables x_1, ..., x_n come from the input space of the data used for learning *Model F*, i.e. relations from the NYT corpus and Freebase, while the output variables come from the columns of the matrix reconstructed by *Model F* shown in Figure 4.3.

The algorithm for inducing a CART tree is described in Section 2.2.3. However, this greedy approach of growing a decision tree implies two main problems, namely *data fragmentation* (overfitting) and *poor probability estimation*. The problem of data fragmentation arises when the splits of the data are too small. Each node added to the hierarchy implies a split on the training set, and consequently, this imposes a margin in the input space; in the worst situation we would fragment the input space as fine as to demarcate each training instance. On the other hand, the problem of poor probability estimation arises as an effect of data fragmentation since the number of instances falling at each leaf diminishes with the depth of the leaf, leading to a poor probability estimation model. Possible solutions to both problems are either applying a Laplacian smoothing in each leaf (Provost and Domingos, 2003), constraining the depth of the tree, add a stopping criterion when the number of instances at split s_i has reached a minimum number of training instances, or add another stopping criterion when the information gain at split s_i is near zero.

We use both of the stopping criterions described above. We set $\beta = 10$ (threshold manually chosen) as the minimum number of instances at any split s_i. We decided not to use tree depth as a heuristic given that most of the times the trees induced tend to be *shallow* (the smallest one has depth $d = 2$, the largest one has depth $d = 13$, and in average the depth is $\overline{d} = 5.57$); for the same reason we decided

not to prune the tree after learning.

4.7.3.3 Learning Bayesian Network Trees

Learning a Bayesian network of unbounded indegree[22] is an NP-complete problem (Chickering, 1996). Thus we opt for a method[23] that provides an optimal DAG in polynomial time ($O(n^2)$) at the cost of expressiveness.[24] More concretely, we assume a tree structure for our proxy model where one node can have several children but only one parent. Choosing a tree structure alleviates both the search in the DAG space and the problem of overfitting since the structure is sparse. And also relevant is the fact that this structure only keeps the links between nodes with highest *weight*.

Since the structure of our proxy model is already known, we are left to learn both the configuration of each *family* (sub-trees) in the DAG and the parameters for such families (local conditional probability functions.) In order to learn the building blocks of each family, i.e. the link between two random variables x_i and x_j (both variables being columns of the matrix reconstructed by *Model F*), we compute their mutual information in order to measure their statistical dependence (similar to the rule selection procedure in Section 4.7.3.1.) This measure can be seen as the difference between the marginal entropy of x_i, namely $H(x_i)$, with respect to the conditional entropy of x_i given x_j, i.e. $H(x_i|x_j)$. This difference shows how much uncertainty in x_i is left by knowing x_j. If x_i is independent of x_j, then the conditional entropy reduces to the term $H(x_i)$, and thus the mutual information between the two random variables is zero. We perform this computation for each possible pair of variables which leads to a matrix **MI** of mutual information scores (Algorithm 3.)

Once the symmetric matrix of mutual information between all random variables is computed (**MI**), we apply a maximum spanning tree algorithm (Koller and Friedman, 2009) on top of it (Algorithm 4.) To do so, we define two sets of nodes,[25] S and T; the first set corresponds to nodes in the spanning tree, the second set corre-

[22] Any node can have any number of links to other nodes.

[23] We tried heuristic methods, such as hill climbing, to search in the space of DAGs, but neither of these heuristics resulted in suitable Bayesian networks in terms of fidelity.

[24] It is difficult to say how much expressiveness we actually lose by applying our prior assumptions.

[25] A node is the graphical representation of a variable.

sponds to nodes to be added to the tree. First, we arbitrarily select a node to be the root of the tree, remove it from T and add it to S. After that, we use mutual information as the criterion for removing a node from T in order to add it to S: We select the node $t_i \in T$ which has the highest mutual information with any of the nodes $s_j \in S$ according to matrix **MI**. This step is applied until $T = \emptyset$. This algorithm guarantees to build a tree structure where a link between the nodes n_i and n_j represents the maximum mutual information between the variables x_i and x_j.

After we learn the families, we then learn the parameters by means of maximum likelihood estimation[26] across the rows of the matrix of predictions of *Model F*; i.e. we take these rows as instances. Thus each parameter $p(x_i|parent(x_i))$ is shared across instances.

As we mentioned before, an advantage of learning a tree structure is less overfitting of the data. In an unbounded BN, as we add more edges, the mutual information as a global score (aggregating this score across all families) increases, leading to a structure that probably will add dependencies found in the data that are just noise. Besides, as it is pinpointed in (Koller and Friedman, 2009), missing a true link would lead to incorrect independencies, but the DAG learned could generalize better than an unbounded dense DAG where spurious links are added; in this case, spurious dependencies would be present in the BN increasing the number of parameters to be learned and leading to overfitting.

```
D: training data
x_i, x_j: random variables
computeMI(D){
foreach x_i in D do
    foreach x_j in D do
        MI_ij = mutual information(x_i, x_j)
    end
end
}
```

Algorithm 3: Algorithm for computing the matrix of mutual information between each pair of random variables.

[26] We also tried a maximum a posteriori approach by using a Laplacian smoothing, but it did not make a significant difference in terms of fidelity.

MI: matrix of mutual information
S: set of nodes in the tree
T: set of nodes to be added to the tree
BN: Bayesian network tree
n_i, n_j, n_k: nodes corresponding to random variables x_i, x_j, x_k
maximumSpanningTree(**MI**){
$S \leftarrow n_k \in T$ //arbitrary node
$T - n_k$
while $T \neq \emptyset$ **do**
  $\arg\max_{n_i \in T, n_j \in S} \mathbf{MI}[n_i, n_j]$
  $S \leftarrow S \cup n_i$
  $T - n_i$
end
}

Algorithm 4: Algorithm for inducing a Bayesian network tree. The tree structure is induced by means of a maximum spanning algorithm implemented in the function *maximumSpanningTree*.

4.7.4 Measurements and Analyses

We describe the methods to evaluate the proxy models in terms of fidelity and generalization; we also describe how we use the proxy models to interpret predictions from *Model F*. We refrain from evaluating interpretability in a quantitative way since it is not clear how to properly do such evaluation.[27]

4.7.4.1 Measuring Fidelity and Generalization

A feasible way to measure fidelity is to observe the proportion of test instances that a proxy model labels in the same way as the black-box system; i.e., given the same input, to what extent the behavior of the proxy model agrees with that of *Model F*? Previous work has measured fidelity mostly in terms of accuracy (Equation 4.6.) However, this score may show a misleading picture of fidelity due to the high number of negative instances. We use two other alternative metrics to measure fidelity, namely F_1 (Equation 4.7) and precision-recall (PR) curves (Manning et al., 2008).

[27] Previous work has used people to rate different proxy models (Huysmans et al., 2011), or has used metrics based in characteristics of the model, such as the number of parameters (Freitas, 2014); but, on the one hand, using people is out of the scope of our work, and, on the other hand, we believe a qualitative exploration of how the models serve to explain predictions is a more comprehensive evaluation rather than measuring how *big* is the model.

In particular, we use $11-point$ PR-curves; for each level of recall we compute the precision of the proxy models: A perfect performance is observed if precision score is $Precision = 1$ for all levels of recall. F_1 and PR-curves can deal with the problem of having a high number of negative instances in the test set. However, we need to threshold the predictions of the proxy models[28] in order to measure F_1 (and accuracy), which may impose an arbitrary decision-boundary for the proxy models. To ameliorate this problem, we compute F_1 (and accuracy) for several thresholds. On the other hand, PR-curves plot a model's performance across thresholds. Besides, PR-curves show ranking performance of the models, a behavior imposed in the loss function of *Model F*, while F_1 (and accuracy) measures only classification performance.

Similarly, we measure how well the proxy models can generalize to test data. This measure can also be taken as a proof of fidelity: If a proxy model has faithfully capture the decision process of *Model F* then we would expect it to generalize in the same way as *Model F* (make the same mistakes and the same hits.) We evaluate generalization abilities with precision-recall curves.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.6}$$

$$F_1 = 2\left(\frac{Precision * Recall}{Precision + Recall}\right) \tag{4.7}$$

$$Precision = \frac{TP}{TP+FP} \tag{4.8}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.9}$$

Where TP: *true positives*, TN: *true negatives*, FP: *false positives*, FN: *false negatives*.

[28] Except for the logic rules whose predictions are already binary.

4.7.4.2 Using Proxy Models for Explaining Predictions

Here we explain how the proxy models can be used to explain the decision process of *Model F* for a particular thresholded prediction.[29] We define an explanation as the process by which the black-box system arrives to a decision. This process is defined in terms of observed variables for a given entity pair. For example, suppose a pair of entities e_i is seen in the dataset to instantiate relations r_l, r_k, and r_m, i.e. the facts $r_l(e_i)$, $r_k(e_i)$, and $r_m(e_i)$ are observed; let's also suppose that *Model F* predicts the fact $r_j(e_i)$ as a true fact. Then, a proxy model would take the first three relations (r_l, r_k, and r_m) as observed input variables in order to explain the prediction of *Model F*.[30]

Then, based on the observed variables given as input, we try to infer which of them influenced the output variable by building an interpretable relationship between input and output variables; in other words, we seek for the input variables that contribute (play a role) to the prediction observed. So, we aim to use a proxy model to explain a prediction such as $capitalOf(London, England) = True$, a correct prediction, or $reviewMovie(Daniel\_Kahneman, Nobel) = True$, an incorrect prediction, in terms of observed facts given as input.

Logic Rules In the case of logic rules, we explain the prediction of a fact $r_j(e_i)$ by application of modus ponens until the target fact is realized. In other words, given a pair of entities e_i as an argument, we instantiate the antecedent of a rule and by virtue of the implication of the rule we also instantiate the head; i.e., we apply the substitution $[e/e_j]$ in the body of a rule $r_i(e) \rightarrow r_k(e)$ which in turn instantiates the head. The realization of the head ($r_k(e_i)$) will in turn serve to instantiate the body and head of another rule, and so on. Thus, by successive application of this transitivity we expect to instantiate the target relation to be explained, namely $r_j(e_i)$. If the target relation is not realized then we have no explanation for the prediction.

[29] We aim to explain binary predictions rather than explaining how *Model F* arrives to a particular probability value since it considerably reduces the complexity of the task.

[30] Our proxy models need input variables in order to predict the value of the output variable to be explained; that is why wee need to use known facts as observed variables.

Decision Trees In a decision tree, the target fact to be explained is expected to lie in a leaf node. Then, an explanation accounts for the path from the root node to the leaf node containing the target relation. In this way, such an explanation can be interpreted as an *if-then* (implication) rule where the body contains the variables in a conjunctive form and the head corresponds to the target relation, i.e. *if* $r_l \wedge r_k ... \wedge r_m$ *then* r_j, where r_j is the target relation.

Bayesian Network Tree An explanation in a tree-structured Bayesian network accounts for the sub-tree that relates the target relation predicted n_j[31] with the variables that elicited the prediction, i.e. the observed input variables, namely n_l, n_k, n_m.[32] The target variable can be either a child or a parent in the sub-tree, as well as the input variables. Then, an explanation is seen as a set of variables influencing the target variable directly or indirectly in either a forward or a backward way. We note that in a tree-structured Bayesian network, a node has only one parent, thus the explanation is easier to interpret than if using a more complex structure. We also note that we use the parameters of the BN tree in an explanation only to give a sense of how strong is the influence of a variable on another variable since we consider to be difficult to interpret such a flow of information in a descriptive way.[33]

4.8 Experiments and Results

We provide an experimental comparison of the three proxy models proposed. We define our experiments along two dimensions, fidelity (taking generalization as another measure of fidelity), and interpretability. Thus the first set of experiments compare how well each proxy model mimics the predictive behavior of *Model F*. In the second set of experiments we provide explanations of particular predictions in order to qualitatively assess the explanations from each proxy model.

[31] n_j is the node in the Bayesian network that represents the relation r_j

[32] The terms n_l, n_k, and n_m are the names of the nodes in the Bayesian network that represent the relations r_l, r_k, and r_m, respectively.

[33] More concretely, as previous work has shown (Lacave and Díez, 2002; Yap et al., 2008), it is easy to interpret the structure and the conditional probabilities of a Bayesian network, but it is difficult to understand the inference process of the BN; i.e. it is difficult to interpret how the information flows from one variable to another until it reaches the predicted variable at inference time, namely explaining the computation of $p(n_j|n_l,n_k,n_m)$.

4.8.1 Fidelity and Generalization

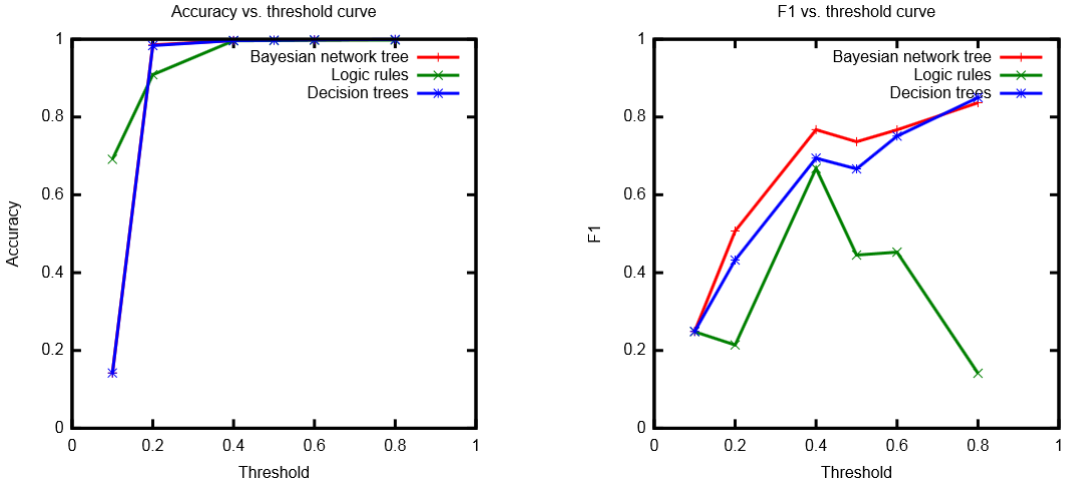
We test how well the proxy models mimic the predictive behavior of *Model F* through a set of 19 test variables shown in Figure 4.3. In Figure 4.4 we observe the performance of the proxy models as classifiers across different threshold values.[34] Figure 4.4a shows that the accuracy of the three models is almost the same, all of them scoring perfectly in most of the threshold values. This metric misleads us since it is heavily influenced by the high number of negative instances; thus, a model predicting everything as negative can achieve an almost perfect score. F_1 scores, shown in Figure 4.4b, is independent of the number of negative instances.[35] We see that at threshold $\gamma = 0.4$ logic rules achieve their best performance, though still not as good as that of either decision tress or the BN tree. We also see that the classification performance of decision trees is always upper-bounded by that of the BN tree, except when $\gamma = 0.8$ when both are almost the same and achieve their pick with a good fidelity score (F_1 just above 0.8). Interestingly, the behavior of both classifiers is very similar through all threshold values.

However, the behavior of the proxy models change drastically when we evaluate them as ranking systems. Figure 4.5 shows performance of proxy models for two thresholds α of *Model F*'s predictions (the ones where the proxy models scored the best.) We note that the behavior of a perfect model, in this type of graphs, would be indicated by a straight line ($Precision = 1$) across all recall values. Such a behavior would indicate that the ranking behavior of the proxy model is identical to that of *Model F*. In both Figures, 4.5a and 4.5b, we clearly observe the superior performance of the Bayesian network tree. Under this ranking measure, decision trees performance is far from the BN tree's performance; and even farther is the performance of logic rules. Specially, in Figure 4.5b we see that the fidelity of the BN tree is close to optimal, having captured the ranking behavior of *Model F*.

We also tested the ranking behavior of the proxy models for generalizing to test instances. Figure 4.6 shows the precision-recall curves of proxy models and *Model*

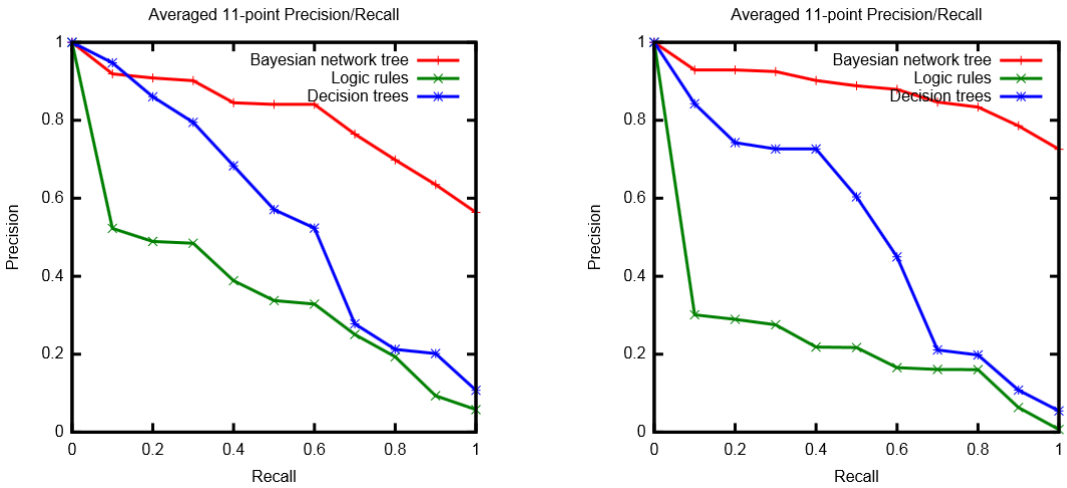
[34]These threshold values are applied to the predictions of the proxy models. The predictions of *Model F* were thresholded at $\alpha = 0.5$.

[35]The predictions of *Model F* were thresholded at $\alpha = 0.5$.



(a) Accuracy score **(b)** F1 score

Figure 4.4: Measures of fidelity of proxy models from a classification perspective: Accuracy and F1 scores.



(a) Predictions of *Model F* thresholded at α=0.4 **(b)** Predictions of *Model F* thresholded at α=0.5

Figure 4.5: Measure of fidelity of proxy models from a ranking perspective: Precision-recall curves.

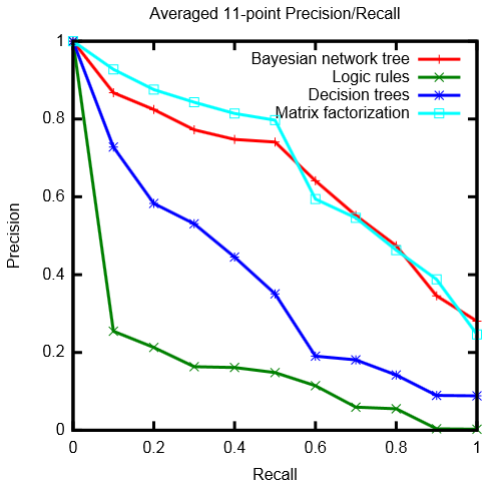


Figure 4.6: Generalization performance of *Model F* and proxy models on test data.

F. We see that only the BN tree has a behavior almost identical to that of *Model F*, specially at recall levels higher than $Recall = 0.5$ where both behaviors are almost indistinguishable. Decision trees and logic rules fall behind by a huge gap.

4.8.2 Interpretability

We show some explanations of how the *Model F* arrived to a particular prediction in terms of our proxy models.

Explanation 1 We first consider the explanation of the prediction showed in Section 4.1, $reviewMovie(Daniel\_Kahneman, Nobel) = True$; this prediction is incorrect since this is a false fact. For this prediction, there are two true facts in the data that we use as observed input values for our proxy models, namely $receive(Daniel\_Kahneman, Nobel)$ and $laureate(Daniel\_Kahneman, Nobel)$. To explain how *Model F* got wrong this instance, we can only resort to the BN tree. Given that the relation *reviewMovie* was not one of the 19 Freebase target variables (Figure 4.3), no decision tree was learned for this variable. As for the logic rules, we could not find an explanation because neither the input nor the predicted relations appeared in the set of logic rules learned; this may be due to the low statistical dependence of these relations with respect to any other relation.

Figure 4.7a shows the active paths of influence from the observed input vari-

ables (*receive* and *laureate*) to the predicted variable (*reviewMovie*) in the BN tree. In this explanation, we observe two types of flow of information. First, we see an *evidential* flow[36] where the observed variable *receive* influences its parent, *nominateFor*, which in turn may influence its parent, *winner*. Similarly, *laureate* influences in a backward way its parent, *award*, which in turn influences its parent, *win*. Second, we also see a *causal* flow[37] where the variable *win* seems to influence the variable *winner* which in turn effects on the predicted variable *reviewMovie*. Probably, both of the observed variables contribute to the prediction of *reviewMoview*, as shown in the conditional probability distribution in Figure 4.7b. However, in order to properly compute inferences in each variable (via Bayes theorem) in a step-by-step fashion we would need more information (marginal probabilities.)

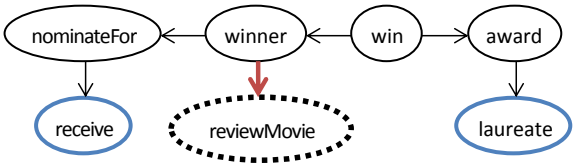
From this explanation we can pinpoint the possible source of error for predicting that Daniel Kahneman is the reviewer of a movie. The variable *winner* seems to have a high correlation with the predicted variable. It seems that *Model F* incorrectly associated the concept of being a winner with that of being the reviewer of a movie. This spurious association was captured by the BN tree in the form of a link. If we observe the rest of the sub-tree, the links provide a sense of direct, or reversed, entailment. For example, the link from *award* to *laureate* can be interpreted as the following entailment: If someone is awarded a price then this person is laureate with that price. In this sub-tree, the only link that seems to be out of context is the one responsible for the incorrect prediction, namely the link from *winner* to *reviewMovie*.

Explanation 2 We explain the following prediction of *Model F*: *arenaStadium* (*Philadelphia_Eagles*, *Canton*) = *True*, an incorrect prediction.[38] In order to explain it, the proxy models use the observed fact *playAt*(*Philadelphia_Eagles*, *Canton*) as input. In the case of logic rules, we cannot explain this prediction since we are not able to recover any rule where the input relates, directly or in-

[36] See Figure 2.7c for an explanation of this type of structure.

[37] See Figure 2.7a for an explanation of this type of structure.

[38] The arena stadium of the Philadelphia Eagles is not the Canton stadium but the Lincoln Financial Field stadium.



(a) Variable predicted: *reviewMovie*.

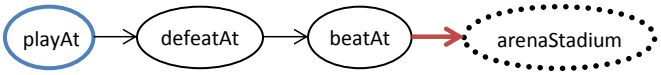
A:parent→B:child	$p(B=1\|A=1)$	$p(B=0\|A=0)$
winner→reviewMovie	0.8119	0.9922
winner→nominateFor	0.8329	0.9951
nominateFor→receive	0.6261	0.9925
win→winner	0.8583	0.9994
win→award	0.7428	0.9982
award→laureate	0.7258	0.9980

(b) Conditional probability distribution.

Figure 4.7: Explanation for the prediction $reviewMovie(Daniel\_Kahneman, Nobel) = True$ using a Bayesian network tree. (a) Excerpt from the sub-graph that spans local influences from the observed variables to the predicted variable. Blue circle indicates observed variable, red arrow indicates a wrong influence over the variable predicted, denoted by a dotted circle. (b) Conditional probability table of the sub-graph.

directly, to the predicted variable. On the other hand, an explanation from a decision tree is the following path (converted to a logic rule): $If\ playAt = True\ then\ arenaStadium = True$. This proxy model learned a single, direct relation from the input to the predicted variable which shows a probable rationale for *Model F* making such an incorrect prediction; it seems that *Model F* believes that if a team T plays in a stadium X then X is the arena stadium of T.

The decision process shown by the decision tree seems to be supported by the BN tree; but the explanation from the latter proxy model shows a more fine-grained decision process. In Figure 4.8a we see the sub-tree where the observed input variable *playAt* is indirectly correlated with the predicted variable *arenaStadium* in a *causal* way. We can see the trail of local influences between these two variables. This trail of influence denotes the decision process that *Model F* may have taken to predict as shown above. To back-up this explanation, in Figure 4.8b we can see the local confidences for propagating the belief from the observed variable to the



(a) Variable predicted: *arenaStadium*.

A:parent→B:child	$p(B=1\|A=1)$	$p(B=0\|A=0)$
playAt→defeatAt	0.8651	0.9978
defeatAt→beatAt	0.8435	0.9999
beatAt→arenaStadium	0.8186	0.9989

(b) Conditional probability distribution.

Figure 4.8: Explanation for the prediction $arenaStadium(Philadelphia\_Eagles, Canton) = True$ using a Bayesian network tree. (a) Excerpt of the sub-graph that spans local influences from the observed variables to the predicted variable. Blue circle indicates observed variable, red arrow indicates a wrong influence over the variable predicted, denoted by a dotted circle. (b) Conditional probability table of the sub-graph.

predicted variable. The probability of instantiating the variable *defeatAt* given our input variable *playAt* is high, namely $p(defeatAt = True|playAt = True) = 0.8651$; in turn, the probability of instantiating the following variable in the path, *beatAt*, given the realization of its parent, *defeatAt*, is also high, namely $p = 0.8435$. Finally, in the last step of the decision process, the probability of instantiating the predicted variable given both the path of influences just described and the likelihood of *arenaStadium* being influenced by its parent is the multiplication of all the conditional probabilities; this computation results in $p = 0.59$ which when thresholded at $\gamma = 0.5$ gives us a $True$ prediction value.

Thus, according to the explanation from the BN tree, it seems that *Model F* learned the series of correlations shown above. We note that at least the final step in the decision process is incorrect. A team T beating some other team at a stadium X does not imply that X is the arena stadium of T; thus there is a spurious entailment learned by *Model F*. Comparing the explanation from the decision tree and the one from the BN tree, we see a difference in the granularity of the explanation; the one from the DT is coarser than the one from the BN tree: The decision process of *Model F* is reduced to a single logical entailment, cutting off a multi-step process.

Explanation 3 Another incorrect prediction by *Model F* is *placeOfDeath* (*Ryutaro_Hashimoto*, *Tokyo*) = *False* where the observed fact taken as input to the proxy models is *meetIn*(*Ryutaro_Hashimoto*, *Tokyo*).[39] In the case of logic rules, the predicate *meetIn* does not appear in any rule due to its low mutual information with any other predicate, thus we have no explanation from this proxy model. We also lack an explanation from the corresponding decision tree since there is no branch in the tree that connects the observed input and the predicted variable; this is because the input variable has low correlation with the target variable and thus it does not appear in any path of the tree. According to the BN tree, the reason for *Model F*'s incorrect prediction is because the degree of separation between both input and output variables is so big (there are more than 30 nodes in between them) that the influence from the input variable vanishes at some point and does not reach the target variable. Therefore, the probability at the target variable, *placeOfDeath*, is near zero, which is interpreted as a *False* value.

Explanation 4 We present an explanation only from the perspective of a decision tree. Consider *Model F*'s prediction *personCompany*(*Michael_Lynton*, *Penguin*) = *True*.[40] The observed fact in the data is *executiveOf*(*Michael_Lynton*, *Penguin*) which we use as an input for the decision tree learned. In Figure 4.9 we can see a hierarchical structure of decisions that classify whether the pair of entities (*Michael_Lynton*, *Penguin*), when instantiating a variable, in this case *personCompany*, results in a true prediction or not. Each decision node is a test of whether the pair of entities is true under the relation denoted in the node, i.e. we test whether each node becomes a true fact. Each left split ($s \leq 0.5$) in a node means a false realization of the relation (a false fact). Each right split means a true fact. In each leaf node we find a prediction for the target variable. If we see at the root node of the tree we find that it is the input variable; hence, from this node we can then extract the following explanation: *If executiveOf* = *True then personCompany* = *True*.

[39] This fact conveys the information about Ryutaro Hashimoto being present in Tokyo.

[40] This prediction can be interpreted as the fact that Michael Lynton is a member of the company Penguin.

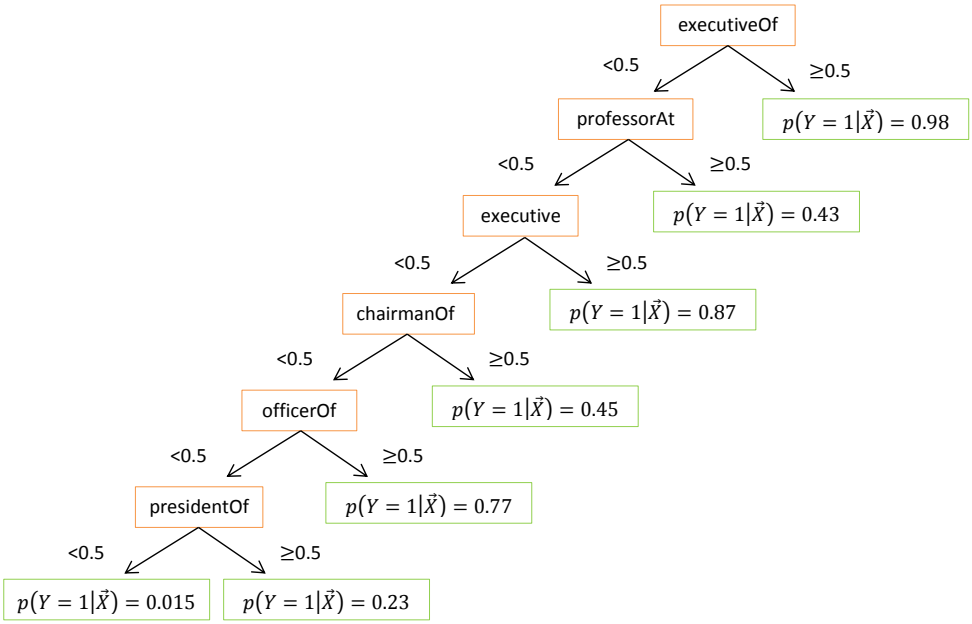


Figure 4.9: Explanation for the prediction $personCompany(Michael\_Lynton, Penguin) = True$ using a decision tree.

Explanation 5 We now present an explanation only from the perspective of the BN tree. *Model F* predicts the fact $restaurantAt(Chilean, Washington) = True$. The observed facts from data are $embassyIn(Chilean, Washington)$ and $diplomatIn(Chilean, Washington)$. Figure 4.10 shows how the influence of the input variables reaches the predicted variable. We can see that *Model F* incorrectly correlated the predicted variable with another variable related to a different context: While *restaurantAt* pertains to a context related to *city life*, the rest of the nodes in the sub-tree pertain to a context related to *politics*.

4.9 Discussions and Conclusions

We first answer our research questions posed in Section 4.3, then we discuss in the following subsections our findings from our experiments. In order to learn the proxy models, we had to conceptualize *Model F* as a multi-label classifier for learning logic rules and a Bayesian network tree (unsupervised learning) and as a binary

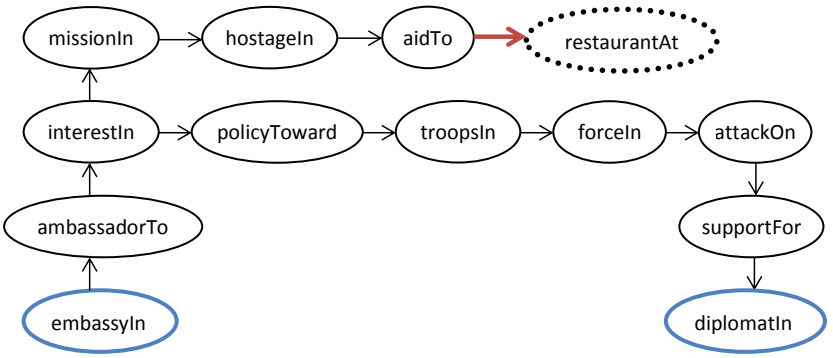


Figure 4.10: Explanation for the prediction $restaurantAt(Chilean, Washington) = True$ using a Bayesian network tree: Excerpt of the sub-graph that spans local influences from the observed variables to the predicted variable. Blue circle indicates observed variable, red arrow indicates a wrong influence over the variable predicted, denoted by a dotted circle.

classifier for learning decision trees (supervised learning). Since *Model F* is a transductive system, it does not learn a function that maps inputs to outputs; however, our proxy models need an input in order to produce an output. Thus, for learning the logic rules and the BN tree, we defined the input and output spaces to be the same, namely all the relations $r_j \in R$ in the domain of *Model F*. This means that these two proxy models can receive as an input a value for any relation in R and predict an output for any relation in R. We used thresholded predictions of *Model F* to populate an unlabeled training set where an instance is formed by the prediction of all relations $r_j \in R$ applied to an entity pair e_i. In the case of decision trees, we defined the input space as all relations $r_j \in R$, and the output space as a single target relation r_k. We selected 19 relations from Freebase as target variables and thus we learned 19 decision trees, one model for each target variable. We used the input space from both the NYT corpus and Freebase relations; these input instances were labeled with predictions of *Model F*.

We evaluated our three proxy models using three metrics, two of them widely used for testing classifiers, namely accuracy and F_1, and precision-recall curves which measure the ranking abilities of systems. We proposed the last metric because *Model F* was trained using a ranking loss function, thus we also wanted to

evaluate to what extent our proxy models learned to rank in the same way as *Model F*. While accuracy scores show a misleading picture of how well the proxy models accomplished their objective, F_1 shows to what extent this models approximated the behavior of *Model F* in terms of classification abilities. In addition, we found that precision-recall curves give another perspective of the behavior of the proxy models –a ranking behavior – and they provide a view that none of the popular metrics from the literature can provide.

Finally, our evaluations show that only one of the proposed models was a good proxy model for *Model F*, namely the Bayesian network tree. It faithfully reproduced the predictive behavior of *Model F* while remaining interpretable for a human. In addition, learning the BN tree took polynomial time. Furthermore, explaining predictions of *Model F* via the BN tree allowed us to see in a graphical way where *Model F* possibly made a mistake and how the inputs connected to the output. Thus, the BN tree, our proposed model, was the only model to provide faithful explanations of the decision process of *Model F*, where inputs are related to outputs in an understandable way.

4.9.1 On The Interpretability-Fidelity Trade-Off of Proxy Models

We found that the Bayesian network tree, as a descriptive model for *Model F*, represents a near-optimal point in the fidelity-interpretability trade-off: It faithfully reproduces (mimics) the ranking behavior of *Model F* while its tree structure is simple to read –a sequence of local influences from an observed input variable to a target variable. As a further test of fidelity we measured generalization abilities; we saw that the BN tree behavior is very similar to that of *Model F* on test data according to our precision-recall curves, meaning that the BN tree may make both similar errors and hits as the black-box system. These two fidelity tests show strong evidence for the BN tree to be an equivalent of *Model F*; i.e. the behavior of the BN tree is *functionally similar* to the behavior of *Model F*.[41] That is, the BN tree

[41] We borrow the term *functionally similar*, when talking about behavior, from studies in behavior and cognition in animals (Shettleworth and Sutton, 2006; Hampton, 2001). In these studies, it is

exhibits functional properties very similar (faithful) to those of *Model F*, such as classification and ranking abilities; thus, in this way, we claim that the BN tree has captured part of the knowledge encoded in *Model F* which is represented in a probabilistic and graphical form.[42]

Also, according to our results, the ranking performance of decision trees and logic rules is not as good as that of the Bayesian network tree. Even though the fidelity of the decision trees is clearly higher than that of the logic rules, it is not comparable to the fidelity of the BN tree as shown by the precision-recall curves. However, in terms of classification performance, we saw that the behavior of the decision trees is close to that of the BN tree, even though the latter still outperforms the former. But, a faithful proxy model, such as the BN tree, should be faithful in both aspects, not only one. Thus we conclude that despite the interpretable symbolic representation that decision trees and logic rules offer they do not represent a good compromise in the fidelity-interpretability trade-off for *Model F*.

A possible reason for the logic rules not performing well in terms of fidelity (aside their brittle nature and consequent inability to deliver probability estimates) is the restrictions we imposed in their structure. We used Horn clauses with only one predicate in the body, and we discarded those relations that were not statistically correlated, by some threshold, to any other relation. It is possible that these restrictions impeded logic rules, to some extent, to better capture the predictive behavior of *Model F*. However, significant improvements in fidelity by using less restricted logic rules are unclear, since the task of ranking seems to be correlated with obtaining probability estimates. Besides, learning unrestricted-size logic rules is an NP-hard problem and resorting to heuristics may be needed.

---

intended to discover any parallels in cognitive capacities between humans and animals; for example, being able to consciously retrieve items from memory. To do so, researchers design experiments where they can examine the behavior of animals and humans in a task that requires the target capacity; if the behavior of the animals is functionally similar to that of humans then this serves as evidence for the animals to have some sort of cognitive capacity similar to that of the humans.

[42] We do not claim that *Model F*'s knowledge is necessarily in the form of a Bayesian network; we also do not claim that the decision process of *Model F* is in the form of a sequence of probabilistic entailments (as that of a Bayesian network). We claim that our BN tree, along with its inference process, is a plausible model to explain, to some extent, the knowledge encoded in *Model F* and its decision process.

Decision trees, on the other hand, have been proven to be poor ranking models (Provost and Domingos, 2003). This seems to be a possible reason for their low fidelity. Even though we tried some heuristics to improve their ranking behavior, such as Laplacian smoothing as recommended in (Provost and Domingos, 2003), these efforts did not allow to fully capture the ranking behavior of *Model F*. Further improving this ability, as in other previous works (Ling and Yan, 2003; Margineantu and Dietterich, 2003), could help to obtain better fidelity scores. We leave this improvement for future work. Nevertheless, decision trees have a disadvantage against a Bayesian network tree. Learning all the knowledge in *Model F* would require learning one decision tree for each relation $r_j \in R$; *Model F*'s domain contain around 4000 relations, which means that we would require the same number of decision trees. On the other hand, a single BN tree can capture all such knowledge; a BN tree requires n nodes to encode the n relations handled by *Model F* (one node per relation) and $n-1$ links to connect the nodes (a compact model.)

4.9.2 On Explanations of Predictions of *Model F*

We explained some of *Model F*'s predictions using the descriptive models learned. In Explanation 2, we saw that the explanation from the BN tree was more fine-detailed than that from the decision tree, i.e. the BN tree captured more steps in the decision process by showing a multi-step explanation, compared to the one-step explanation of the decision tree. On the other hand, we found no explanation from the logic rules for any of the proposed predictions because either the observed variables or the predicted variables, or both, were not found in the set of rules due to their low mutual information with any other variable, possibly a drawback of our rule induction algorithm. The BN tree was able to explain all the wrong predictions made by *Model F* in the form of probabilistic entailments. The observed variables influenced intermediate variables which in turn influenced other variables until the probabilistic influence reached the predicted variable. Each entailment can be seen as a basic step in such process, where a spurious link from one of the observed or intermediate nodes towards the predicted variable can explain why *Model F* predicted as such; i.e., the BN tree represents in a graphical way both the correct and

the spurious correlations learned by the black-box.

The explanations mentioned above were obtained for a specific type of prediction, namely those where *Model F* wrongly predicts as true a target fact. But, we also showed an explanation for another type of wrong prediction, that where the target fact is wrongly predicted as false. In the explanation given by the BN tree, we could understand why *Model F* predicted as such. The degree of separation between observed and predicted nodes was so big that the influence from the former node to the latter became negligible; thus the conditional probability of the predicted node was near zero, meaning a false value for the prediction.

One possible drawback of our approach is the lack of tests on users in order to collect evidence of how useful are the BN tree explanations for other people. Even though we have based our work in the findings of psychological experiments of previous works (Huysmans et al., 2011; Pacer et al., 2013), we believe that performing evaluation tests on users may be useful, which we leave for future work.

As another possible research question to answer in future work, we think that it is not totally clear what is a suitable level of granularity for an explanation induced from a BN tree. In our current form of explanation, a structural one, we only show the local influences from the observed variables to the predicted variable in order to keep simplicity; however, we may lose transparency by not using the local conditional probability distributions and not showing how the beliefs are propagated through the nodes (i.e. the *reasoning* process of the Bayesian network (Nielsen et al., 2008).) On the other hand, by showing these parameters and this process we may add an undesirable complexity in the explanation if the user is not familiar with Bayesian networks; we leave this as an open research question for future work.

4.9.3 Final Remarks

In this chapter, we showed that our BN tree served as an equivalent model of *Model F* since the classification and ranking behaviors associated with the latter are faithfully found in the former according to our experimental framework. Nevertheless, we do not claim that the BN tree is a *replica* (with a different representation) of the knowledge encoded in *Model F*; it rather is a plausible model (an approximation)

of what *Model F* has learned. When the BN tree is attached with its inference algorithm, then we claim that we have obtained a plausible model of how *Model F* achieves its predictions; and again, we do not claim that *Model F*'s decision process is based on a sequence of probabilistic entailments; this entailment process, along with the BN tree, is a model that turns out to fit very well the behavioral data coming from *Model F*, faithfully mimicking its abilities, and as such, it becomes a *functionally similar* model of *Model F*.[43]

Also, we tied the task of interpretability with Marr's analysis at the representation and algorithmic level by showing how an interpretable proxy model provides a similar explanation to that obtained by Marr's analysis. By doing so, we provided another view of the task of explaining predictions of a black-box system, namely as that of explaining the decision process of a black-box system. This high-level reconceptualization allows us to better situate this task by finding a parallel with other field of science, namely cognitive science. In this way, we hope future research in interpretability will be further inspired by methods in cognitive science.

[43] We also note that other interpretable models may fit even better the behavioral data coming from *Model F*, thus serving as a better approximation to *Model F*'s decision process.

Chapter 5

Behavior Analysis of *ESIM*, *DAM*, and *CE*: Evaluating Robustness

5.1 Introduction

The task of Natural Language Inference (NLI) (also known as Recognizing Textual Entailment) has received a lot of attention due to its complexity: given two sentences, called premise and hypothesis, a system has to categorize their relation into three classes, namely *entailment* (the information in the hypothesis is true given the information in the premise), *neutral* (the information in the hypothesis may be true given the information in the premise), and *contradiction* (the information in the hypothesis either contradicts or has nothing to do with the information in the premise). This task has elicited state-of-the-art models which have achieved impressive results, such as the *ESIM* system (Chen et al., 2017) which has scored close to what seems to be the ceiling score of the Stanford NLI (SNLI) dataset (Bowman et al., 2015). These results are impressive due to both the linguistic abilities and knowledge required to solve the task of natural language understanding (LoBue and Yates, 2011; Maccartney, 2009). For example, NLI systems may need to know about lexical semantic relations, such as hypernymy and antonymy,[1] asymmetric

[1]For example, knowing that a cat is a type of vertebrate, and vertebrate is the opposite to invertebrate.

relationships,[2] and causality,[3] among other types of knowledge and abilities.

With such positive results, new systems improving accuracy results are continuously proposed, leading closer to a state where the SNLI dataset is finally solved. However, the ever-growing complexity of such models prevents people from fully understanding the phenomena being captured by such models. As a direct consequence, validating that the systems have captured the abilities and knowledge required for the task of NLI, such as those described above, relies on a single signal from the test set; this signal may not provide enough evidence to prove, in a fine-grained detail, what the systems have learned. Similarly, evaluating whether the systems have captured a bias, have been influenced by certain factors, or have captured any possible abnormal behaviors becomes a difficult task just by looking at test accuracy (Kummerfeld et al., 2012; Sammons et al., 2010).

Previous work in the natural language processing (NLP) community has evaluated abilities of complex systems (Isabelle et al., 2017; B. Hashemi and Hwa, 2016; White et al., 2017), it has also evaluated the robustness of systems facing adversarial instances (Jia and Liang, 2017), and it has identified biases learned by ReLe systems (Zhao et al., 2017; Bolukbasi et al., 2016).[4] For example, in (Jia and Liang, 2017) ReLe systems were trained to find answers in a text given a query. These systems were able to extract the correct answer with a high test accuracy; however, when Jia and Liang (2017) added an adversarial (spurious) sentence in the text which overlapped in content to the sentence where the correct answer was to be found, the system got *confused* and extracted the answer from the adversarial sentence rather than from the correct sentence.[5] In this way, Jia and Liang (2017) exposed the lack of robustness of these ReLe systems to alterations in the text. More concretely, they showed that the predictions of the systems were based in certain patterns that when

[2]For example, knowing that a cat is a type of vertebrate but a vertebrate is not necessarily a cat.

[3]For instance, knowing that if an object is thrown to the air and there is no obstacle in between then it will fall down.

[4]We refer to Section 3.2.2 for a deeper description of previous work.

[5]For example, given the query *Where did the person A was born?* and a text (not shown here) where the correct answer is contain in the sentence *Person A, whose parents come from France, was born in 1990 in London, England.*, Jia and Liang (2017) then add an adversarial sentence such as *Person B was born in New York.*. When the system receives the query, it answers it with the string *New York* instead of *London* which is the correct answer.

arranged in a certain configuration they could easily mislead the system, i.e. the systems' predictions were influenced by some confounding factors.

However, analyzing the robustness[6] of NLP ReLe systems is in its early stages. In this chapter, we aim to advance in this line of research by studying the robustness of three ReLe systems for the task of natural language inference, namely the *ESIM* system, one of the best systems to date on the SNLI dataset, *DAM* a former state-of-the-art system, and *CE*, a basic ReLe system. Similar to work in adversarial instances, we propose to create challenging instances that only systems with a good grasp of natural language would classify correctly; thus while these instances may be challenging for a ReLe system, they are easy for a human. In this way, we propose to study how well *ESIM*, *DAM*, and *CE* react to a transformation in the input space that yields challenging instances, i.e. we obtain challenging instances from existing instances after we apply our transformation. This transformation consists in swapping two words, one word from the premise sentence with one word from the hypothesis sentence; this simple alteration may cause the class label of the transformed instance to change.

Let's take the instance in Example 5.1 to show our transformation. This instance is a *contradiction*, the information in the premise sentence p contradicts the information in the hypothesis sentence h;[7] if we swap the word pair *(elderly, young)* we yield a new instance, shown in Example 5.2, where the semantics of each sentence has changed, to some extent, but the class label remains the same, a *contradiction*. The preservation of the class label is because we swapped an antonym word pair, and since antonymy is a symmetric relationship, swapping two antonyms does not affect their relation, and since the word pair remains in an antonymy relation then we did not affect the relationship of the premise and the hypothesis. Thus, if a system classifies correctly the instance in Example 5.1 and is able to handle antonymy, then we would expect it to correctly classify the transformed instance as well.

(5.1) p : An elderly woman sitting on a bench.

[6] We refer to Section 3.2.2 for a description of robustness.

[7] In Section 5.6.1 we describe in detail the task of natural language inference.

h : A young mother sits down.

(5.2) p : An young woman sitting on a bench.

h : A elderly mother sits down.

At a first glance, it seems that if a system correctly classifies our transformed instances, despite the changes introduced with respect to the original instances undergoing the transformation, then this system would be deemed as robust. For example, if a system correctly classifies the instances in Examples 5.1 and 5.2 as *contradiction* then it would seem that the system has learned antonymy. However, this picture may be misleading. Given that the system got the original instance correct, maybe it got the transformed instance correct as well just because the two instances closely resemble each other; i.e., probably the system did not find the transformation to be so significant as to change the class label. In other words, a confounding factor probably affected the behavior of the system, namely the system's *insensitivity* to changes in the input data. Another possible confounding factor explaining why the system classified correctly the transformed instance is because the word pair *(young, elderly)* (where *young* is in the premise and *elderly* is in the hypothesis) appeared in many training instances of class *contradiction* and thus the system learned that any instance containing this word pair is likely to be a *contradiction*, regardless of the words surrounding the word pair in both premise and hypothesis sentences. In other words, we say that the word pair is *polarized*, like a magnet that has negative/positive polarity, the word pair may have a *contradiction*, *entailment*, or *neutral* polarity according to the class of instances it was mainly seen in the training set; but opposite to a magnet, a polarized word pair *attracts* the same polarity, i.e. influences the system to predict the same class label as its polarity.

Now, what if the system actually gets the original instance correct but it gets the transformed instance incorrect? Can we attribute this result to a lack in robustness? Not necessarily, as another confounding factor may play a role. It may be the case that the word pair (*young, elderly*) in the transformed instance was never seen at training time and so the system was not able to handle something it did not learn about. This scenario seems to be misleading again since we are evaluating the

system on an *unseen word pair* for which there was no training signal and thus no knowledge was gained by the system about the interaction of *young* with *elderly*.

This thought experiment leaves us then with some questions, such as to what extent the system is actually robust? How do we measure its robustness? How can we validate that the impressive, or unimpressive, behavior that we observe is due to the abilities of the system and not due to confounding factors, such as *insensitivity*, *polarity*, and *unseen word pairs*? What other confounding factors may affect a system trained on the SNLI dataset?

Studying the robustness of a system thus implies analyzing what factors may affect a system's behavior, so that we are sure we attribute a robust behavior to the abilities of the system rather than to confounding factors. Then, in order to study the robustness of our target ReLe systems, we draw motivation from behavioral science. In this discipline, the aim is to analyze the behavior of a subject mainly for two reasons; first, to understanding *environment-behavior* relationships (Epling and Pierce, 1986), i.e. how environmental factors influence the behavior of the subject under study; and second, to discover possible abnormal behaviors (Birkett and Newton-Fisher, 2011). These analyses of behavior are performed under controlled scenarios in order to discover or rule out any possible confounding factors; this experimental setting then allows the researcher to statistically validate that the behavior observed is due to the target variable under analysis and not due to a confounding factor. In other words, this experimental setting is said to provide *internal validity* to the results obtained.[8] Motivated by the research from this discipline, we borrow both research questions and design of experiments in order to elaborate a framework to systematically analyze the robustness of ReLe systems to cope with our proposed transformations in the input data.

In order to provide internal validity in our study, we control for possible confounding factors such as the length of the sentences and the presence of words out of vocabulary; however, we recognize the presence of three factors which we call *insensitivity*, *polarity*, and *unseen pairs* that we cannot control by keeping them

[8] We refer to Section 3.2.1 for a description of internal validity and an example of previous work in the behavioral science.

identical for both groups control and treatment, so we control them in a statistical way. Thus, we analyze how these three target factors influence the behavior of the ReLe systems. Based on preliminary observations, we hypothesize that these factors systematically influence the robustness of the systems. Finally, we ask whether these factors affect in a similar way the *ESIM*, *DAM*, and *CE* systems. Do the state-of-the-art system, *ESIM*, is less prone to be affected by such factors? Is the test accuracy of *ESIM* correlated with its robustness? Are there any behavioral patterns in the three ReLe systems?

5.2 Problem Definition

We aim to evaluate the robustness of ReLe systems trained on the SNLI dataset (SNLI systems from now on.) We define the problem as follows: Given a pre-trained SNLI system which receive as input an instance $\mathbf{x}$ of the form (p,h), where p is a premise sentence and h is a hypothesis sentence, and the system responds with a class label $y \in \{$ *contradiction*, *entailment*, *neutral*$\}$, we aim to evaluate its ability to classify instances transformed by our operation T which consist in swapping word pairs. This evaluation is achieved by observing its behavior on the transformed instances while statistically accounting for confounding factors that may influence such a behavior, namely *insensitivity*, *polarity*, and *unseen word pairs*.

5.3 Research Questions and Hypotheses

Considering the three target SNLI systems *ESIM*, *DAM*, and *CE*, our transformation T, and the confounding factors that we call *insensitivity*, *polarity*, and *unseen word pairs* we guide our research with the following questions and hypotheses:

Research Questions

1. Are the SNLI systems robust on our transformed instances?
    - Do they obtain a similar accuracy score on our transformed instances as that obtained on SNLI development data?
    - In the case that they achieve high accuracy scores on our transformed instances, is it due to their robustness or due to confounding factors?

- Is their predictive behavior affected by the confounding factors listed above?

2. Is there any relation between a system's accuracy score on the SNLI development data and the way its behavior is influenced by the confounding factors? I.e., is the best scoring system on SNLI development data less prone to have its score on transformed instances affected by the confounding factors?

3. Are there any common behavioral patterns embodied by the systems? I.e. Do different systems are affected by different confounding factors?

Hypotheses

1. We hypothesize that part of the impressive accuracy scores that we observe on SNLI test data are due to one or both confounding factors, namely *insensitivity* and *polarity*; i.e., these two factors contribute to the performance of the systems on test data.

2. We also hypothesize that *unseen word pairs* systematically affect the performance of the systems in a detrimental way; i.e. drop in performance is correlated with having unseen word pairs in transformed instances.

5.4 Contributions

- We analyze the robustness of three SNLI systems (two of them widely popular) based on methods from behavioral science where we control for confounding factors and statistically analyze the effects of other factors in the response of the systems.

- We propose a simple transformation on instances that yields challenging instances useful to test the robustness of systems, namely a swap of word pairs. Also, we manually annotate the transformed instances to guarantee correct labelling of the instances. Furthermore, we will release this new test set of transformed instances so that the community can evaluate new SNLI systems.

5.5 Scope and Limitations

We aim to understand how three specific ReLe systems trained on the SNLI dataset performed on challenging instances (we obtained these instances via a transformation on SNLI instances) by using methods from behavioral science. These methods allow us to control for certain confounding factors and to statistically analyze other confounding factors without the need to open the systems under study; i.e., all the analyzes are done just by looking at stimulus-response (input-output) patterns. Hence, we do not aim to provide any insight into the inner workings of the systems; the methodology used is designed to ignore what happens inside the systems without compromising the validity of the results obtained.

Even though this methodology allow us to control for confounding factors, it is difficult to control for all possible factors either because there are hidden factors which we are not aware of, because controlling for such factors would lead us to extremely few data to test the systems, or simply because it becomes unfeasible to do so. Hence, there are factors for which we were not able to control. In addition, in our analyzes we investigate each of the confounding factors independently; however, confounding factors may interact with each other.

Another limitation is the generalization of results obtained on the transformed instances. These instances are just one type of challenging instances; i.e., other transformations may yield other types of challenging instances which may rise other confounding factors. Furthermore, after controlling for confounding factors, we obtain samples in the order of 620 transformed instances, a relatively small sample size. Thus, our results provide a piece of evidence for the robustness of the systems under study rather than a fully conclusive view of their robustness.

Furthermore, our aim in this chapter is to provide an evaluation and analyses that may be useful for future work on improving the SNLI dataset or the systems. We do not aim to carry out any improvements in this chapter, but rather to provide explanations of why the systems are robust or not.

5.6 Systems Under Study

In this section we provide descriptions of the task of natural language inference, the SNLI dataset, and the systems that address this task.

5.6.1 Natural Language Inference

This task, also known as Recognizing Textual Entailment (RTE) (Dagan and Glickman, 2004; Dagan et al., 2009), requires systems to classify two sentences under three possible classes, according to the relation between the two. We call these sentences premise and hypothesis, and the possible classes are *entailment*, *neutral*, and *contradiction*. The sentences fall under the *entailment* class if the information in the hypothesis entails the information in the premise; i.e., the information in the hypothesis must be true given the information in the premise. Consider the pair of sentences in Example 5.3; we see that the hypothesis *h* follows from the premise *p* because the expression *two men* is subsumed by the term *people* who are doing the same activity, *riding bicycles*.

On the other hand, in a *neutral* relation, the hypothesis sentence may be true given the premise, but not necessarily. As an example see Example 5.4; the hypothesis *h* provides extra-information not contained in the premise sentence *p*. Thus, we cannot conclude that the hypothesis follows the premise, though it may follow due to the high overlap of information; i.e., we agree that *two men on bicycles* is equivalent to the expression *people are riding bicycles*, though we are not sure whether the expression *in a race* is also equivalent to the term *on the street* because we do not know if the race takes place on the street.

Finally, we say the hypothesis is in a *contradiction* with the premise if either the information from both is contradictory or unrelated. In Example 5.5 we observe the first type of contradiction, where the two men mentioned in both sentences are doing different things; thus the hypothesis *h* contradicts the premise *p*. In Example 5.6 we see the second type of contradiction, where the hypothesis contains no overlap of information with the premise sentence.

(5.3) *p* : Two men on bicycles competing in a race.

h : People are riding bicycles.

(5.4) p : Two men on bicycles competing in a race.

h : People are riding bicycles on the street.

(5.5) p : Two men on bicycles competing in a race.

h : Two men on bicycles going to the cinema.

(5.6) p : Two men on bicycles competing in a race.

h : Three women are eating rice.

We used the SNLI dataset for training the *ESIM*, *DAM* and *CE* systems which we describe below.

5.6.2 Stanford Natural Language Inference Dataset

The SNLI (Bowman et al., 2015) was created with the purpose of training large-scale systems. It consists of around 570 000 instances in total, divided into training, development, and test sets. It was created in a two-phase process. First, sentences were taken from a pre-existing dataset of images labeled with a description provided by a human; these descriptions served as the premise sentences. Then, the premise sentences were given to Amazon Mechanical Turk workers who were instructed to provide a hypothesis sentence that entails, contradicts, and may entail the premise sentence. In this context, an entailing hypothesis sentence is defined as an alternative, true description of the image from where the premise sentence comes from;[9] thus, a possible way of writing an entailing hypothesis is to paraphrase the premise. A contradictory hypothesis is defined as a false description of the image; one possible way of writing this type of hypothesis is to describe a completely different image. A neutral hypothesis is defined as a possible true description of the image; then, one way of writing such a description is to add extra information to the premise sentence. An instance is then formed by pairing a premise sentence with its corresponding hypothesis sentence in one of the three classes, *entailment*, *contradiction*, and *neutral*. Examples of instances are provided in Examples 5.3, 5.6, and 5.4.

We provide some statistics about the SNLI in Tables 5.1, 5.2, and 5.3. We compare the length of hypothesis sentences against premise sentences in instances

[9]We note that the workers were not given the images, only the descriptions of such images.

Label	Premise	Hypothesis
Neutral	12.84 (5.65)	8.26 (3.39)
Contradiction	12.85 (5.65)	7.36 (2.81)
Entailment	12.84 (5.65)	6.63 (2.73)

Table 5.1: Average sentence length and standard deviation in both premise and hypothesis sentences from the training set.

Label	Premise	Hypothesis
Neutral	13.81 (6.28)	8.34 (3.36)
Contradiction	13.98 (6.34)	7.39 (2.89)
Entailment	14.01 (6.30)	6.81 (2.94)

Table 5.2: Average sentence length and standard deviation in both premise and hypothesis sentences from the development set.

Label	Training set	Dev set
Neutral	4.33	4.36
Contradiction	3.76	3.82
Entailment	4.66	4.96

Table 5.3: Average word overlap in premise and hypothesis sentences per class label in both training and development sets.

in the training set; we observe that the average length of hypothesis sentences, in *entailment* instances, is almost half the length of its corresponding premise. This may be an effect due to either paraphrasing or deleting structures such as prepositional phrases from the premise sentence which may shorten the length of the resulting hypothesis. We make a similar observation in the same class label in the development set. In the case of *neutral* class instances, the length of the hypothesis is also shorter with respect to that of the premise, but not as much as in the *entailment* class. Probably this indicates that the annotators did not just add extra information to the premise, but rather paraphrased it and then added extra information.

5.6.3 CE

Our simplest system is a conditional encoder (*CE*) system. It consists of two bidirectional LSTMs. The first one, reads the premise sentence and encodes it into a sentence embedding, **p**; the second bidirectional LSTM receives as input both the embedding of the premise, **p**, and the hypothesis sentence. It then encodes the hypothesis into the embedding **h**, conditioned on the information stored in **p**. On top of the final hidden state of the second bidirectional LSTM, a *softmax* layer is placed in order to compute a probability distribution over the three classes, *entailment*, *neutral*, and *contradiction*. This system can be considered as a building-block of *ESIM* since both use bidirectional LSTMs, but *ESIM* enhances them with more complex artifacts.

5.6.4 DAM

The *DAM* (Decomposable Attention Model) (Parikh et al., 2016) system consists of 2-layer multi-layer perceptrons (MLPs) factorized in a 3-step process. First, a soft-alignment matrix is created for all the words in both the premise and hypothesis sentences. Then, each word of the premise sentence is paired with the soft-alignment representation of the hypothesis sentence and fed into an MLP, and similarly for each word in the hypothesis with the soft-alignment of the premise sentence. The resulting representations are then aggregated; the vector representations of the premise sentence are summed up and the same for those of the hypothesis sentence; the new representations are then fed to an MLP followed by a linear layer and a softmax whose output is a class label. As a final note, we use pre-trained GloVe embeddings with dimensionality $d = 300$ which are not updated at training time. All layers use ReLU function.

We choose to evaluate robustness of *DAM* since it is a former state-of-the-art system and its accuracy score is not that far from the accuracy score of *ESIM*, though their architectures are different.

5.6.5 ESIM

The *ESIM* (Enhanced Sequential Inference Model) system (Chen et al., 2017) consists of three sub-systems. The first sub-system reads both premise p and hypothesis h sentences using bidirectional LSTMS (see Section 2.1.1.3), where each sentence is mapped to an embedding ($\mathbf{a}$ for p and $\mathbf{b}$ for h.) We note that each word p_i and h_j from both sentences ($p_i \in p$ and $h_j \in h$) is associated with its own embedding ($\mathbf{a}_i$ and $\mathbf{b}_j$ respectively), where this embedding carries the contextual information of the word, i.e. information about the words surrounding the current word.

Then, the second sub-system builds a representation of local information. First, it aligns each possible pair of words (a_i, b_j) from premise and hypothesis sentences; i.e., this sub-system creates a weight w_{ij} for each word pair based on the similarity (dot product) of the embeddings corresponding to the words in the pair. After that, this sub-system computes another embedding for each word $a_i \in p$ and $b_j \in h$, namely $\tilde{\mathbf{a}}_i$ and $\tilde{\mathbf{b}}_j$; this embedding is the weighted sum of the embeddings of the words in the other sentence, as shown in Equations 5.7 and 5.8, where $|p|$ and $|h|$ denote the length of premise and hypothesis sentences, respectively. We can interpret this embedding as the composition of relevant information for the current word. Finally, this sub-system enhances the local information of premise and hypothesis by computing both the difference and the element-wise product of the representation of the sentence and the weighted representation, $\mathbf{a}-\tilde{\mathbf{a}}$, $\mathbf{a}\odot\tilde{\mathbf{a}}$, $\mathbf{b}-\tilde{\mathbf{b}}$, $\mathbf{b}\odot\tilde{\mathbf{b}}$; then, the resulting embeddings are concatenated with the original vectors, as shown in Equations 5.9 and 5.10. The intuition behind this new representation is that it may capture some interactions between embeddings.

$$\tilde{\mathbf{a}}_i = \sum_{j=1}^{|h|} \frac{exp(w_{ij})}{\sum_{k=1}^{|h|} exp(w_{ik})} \tilde{\mathbf{b}}_j \forall i \in [1,...,|p|] \tag{5.7}$$

$$\tilde{\mathbf{b}}_j = \sum_{i=1}^{|p|} \frac{exp(w_{ji})}{\sum_{k=1}^{|p|} exp(w_{ki})} \tilde{\mathbf{a}}_i \forall i \in [1,...,|h|] \tag{5.8}$$

$$\mathbf{m}_a = [\mathbf{a}; \tilde{\mathbf{a}}; \mathbf{a}-\tilde{\mathbf{a}}; \mathbf{a}\odot\tilde{\mathbf{a}}] \tag{5.9}$$

$$\mathbf{m}_b = [\mathbf{b}; \tilde{\mathbf{b}}; \mathbf{b} - \tilde{\mathbf{b}}; \mathbf{b} \odot \tilde{\mathbf{b}}] \tag{5.10}$$

Finally, the third sub-system explores the relationship between premise and hypothesis sentences by composing their latest representations learned, namely $\mathbf{m}_a$ and $\mathbf{m}_b$. This composition takes place via a bidirectional LSTM; after that, both average and max pooling are performed over the latest representation. Finally, the resultant representations from the pooling operations are concatenated to form the final representation. This final embedding is then fed to a *softmax* to compute probabilities for each of the three classes, *entailment*, *neutral*, and *contradiction*. As a final note, we use pre-trained GloVe embeddings with dimensionality $d = 300$ which are updated at training time.

We choose to study this system because is currently the state-of-the-art system on the SNLI dataset achieving an accuracy score close to what seems to be the ceiling accuracy of this dataset.

5.7 Methods and Materials

5.7.1 Data

We describe the data that we use to evaluate the robustness of the *ESIM*, *DAM*, and *CE* systems.

5.7.1.1 Overview of the Whole Process

We evaluate the systems under two experimental conditions, namely *in situ* and *ex situ*.[10] For each experimental condition we generate test sets to evaluate robustness, as shown in Figures 5.1 and 5.2. Each test set comprises two samples (each sample has in average 628 instances), a control sample and a transformed sample, similar to observational studies (Song and Chung, 2010) where control and treatment groups are clearly differentiated. A transformed sample contains challenging instances which we generate from instances already known to the systems, namely

[10]The main distinction between these two experimental conditions is the form of the instances, wihch we will describe in Section 5.7.1.4. Having these two conditions allow to evaluate the robustness of the systems in a finer-grained detail.

the control instances which are sampled from the training data. Then, we feed the systems under study with both samples, control and transformed,[11] and observe their behavior. The purpose to have control and transformed samples in a test set is to measure the effect that a transformation on control data, such as swapping word pairs, has on the systems. If the systems are robust then such a transformation should not play a significant role in the accuracy scores, i.e. the accuracy on both control and transformed samples should be similar. Having control and transformed samples also allow us to statistically analyze whether a confounding factor is affecting the behavior of the systems. In the following sections we describe in more detail the whole process.

Figure 5.1: Process to obtain *in situ* instances. Instance *i* corresponds to a control instance, while *i'* corresponds to a transformed instance.

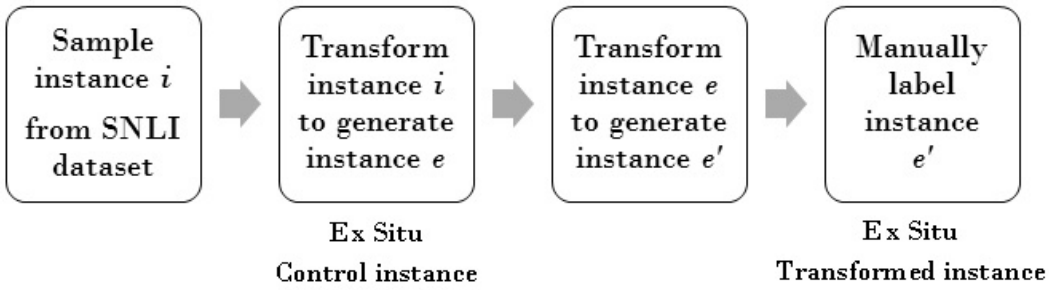


Figure 5.2: Process to obtain *ex situ* instances. Instance *e* corresponds to a control instance, while *e'* corresponds to a transformed instance.

5.7.1.2 Sampling Procedure

We sample instances from the SNLI training set according to a criterion, namely whether the instance contains one of the following types of word pairs: Antonym,

[11] In our case a transformed sample corresponds to a treatment group.

hypernym, or hyponym; example of these types of word pairs are *(elderly, young)*, *(cat, animal)*, *(animal, cat)*, respectively. We use the antonym word pairs from the dataset of Mohammad et al. (2013) and the hypernym-hyponym word pairs from the dataset of Baroni et al. (2012). Thus, for each word pair (w_1, w_2) in the collection of antonym, hypernym, and hyponym word pairs obtained, we look for instances $i = (p, h)$, where p and h correspond to premise and hypothesis sentences respectively, such that $w_1 \in p \ \ \& \ \ w_2 \in h$.[12] We separate those instances containing antonym word pairs from those containing hypernym or hyponym word pairs. We call these samples *control* since they contain instances in their original form without any transformation,[13] and we call them I_A and I_H (see Table 5.7.) The sample I_A contains control instances where each instance contains an antonym word pair. The sample I_H contains control instances where each instance contains a hypernym or a hyponym word pair. Both of these samples will be used in the *in situ* experimental condition.

An example of an instance sampled from the training set is the one shown in Example 5.11. We see that this instance contains the word pair $(sunset, sunrise)$, which is an antonym pair found in the antonym dataset mentioned above, where $w_1 = sunset \in p$ and $w_2 = sunrise \in h$.

(5.11) p : A soccer game occurring at sunset.

h : A basketball game is occurring at sunrise.

5.7.1.3 Transformations

We have three different transformations that we apply on data. Two transformations, namely T_{swap} and T_{sub}, are used to generate transformed samples, i.e. challenging instances to evaluate robustness. The third transformation, T_{ex}, has the purpose to generate *ex situ* instances (control and transformed) from *in situ* instances.

Swapping Word Pairs Once we have collected control samples according to the sampling procedure described in Section 5.7.1.2, we apply our transformation T_{swap}

[12] In the case of antonyms we also look for $w_2 \in p \ \ \& \ \ w_1 \in h$ since antonymy is a symmetric relation.

[13] These control samples are to be used in the *in situ* experimental condition. In order to obtain control samples for the *ex situ* experimental condition we need to transform these *in situ* control samples.

on these instances in order to yield a set of *transformed* instances which are then used for testing the robustness of the target systems.

Transformation T_{swap} targets the word pairs (w_1, w_2) described in the previous section: We swap the position of w_1 with the position of w_2; this transformation yields a transformed instance where now $w_2 \in p$, $w_1 \in h$ and $w_1 \notin p$, $w_2 \notin h$.[14] The class label of the new instance may or may not change with respect to the class label of the control instance undergoing the transformation. For example, take the *contradiction* instance in Example 5.11 as the control instance. After we apply T_{swap} on this instance, we yield the transformed instance shown in Example 5.12; we observe that the class label of this new instance is the same as that of the control instance, *contradiction*, because antonymy is a symmetric relation.

(5.12) p : A soccer game occurring at sunrise.

h : A basketball game is occurring at sunset.

Thus, it depends on the type of word pair swapped whether the class label of the transformed instance changes or not. Instances of class *contradiction* will yield transformed instances with the same class label, though instances of class *entailment* and *neutral* may yield transformed instances with a different label. Consider the control instance of class *entailment* in Example 5.13. After swapping the target word pair $(footbridge, bridge)$ we obtain the transformed instance in Example 5.14. This new instance has a different class label (*neutral*) from that of the control instance from which it was generated (*entailment*); then we say that the class label changed from the control to the transformed instance.

(5.13) p : A little girl hugs her brother on a footbridge in a forest.

h : A pair of siblings are on a bridge.

(5.14) p : A little girl hugs her brother on a bridge in a forest.

h : A pair of siblings are on a footbridge.

T_{swap} when applied on the control sample I_A generates the transformed sample I_{TA1}, and when applied on the control sample I_H generates the transformed sample

[14] If a word w_1 or w_2 appears more than once, we replace all the appearances with its corresponding pair, w_2 or w_1.

I_{TH} (see Table 5.7.)

Substituting Words Even though transformation T_{swap} is our main transformation to generate the transformed instances, we need to use another transformation, namely T_{sub} which instead of swapping word pairs it substitutes a word in a word pair. The reason for using T_{sub} is to obtain transformed instances which have a different class label from that of the control instances. We recall that transformation T_{swap} applied on control instances of class *contradiction* generates transformed instances of the same class; but, we need a change in class label to analyze a particular confounding factor, namely insensitivity (Section 5.7.3.1). Transformation T_{sub} yields transformed instances with different class labels.

T_{sub} replaces one of the words in a word pair by either a synonym, hypernym, or hyponym[15] of the other word. Consider the antonym pair $(elderly, young)$ in the control instance shown in Example 5.15. T_{sub} selects the word *young* and replaces it with *aged*, a synonym of *elderly*; this results in the transformed instance in Example 5.16. We notice that the class label of the two instances are different, *contradiction* and *neutral*, due to T_{sub}.

(5.15) p : An elderly woman sitting on a bench.

h : A young mother sits down.

(5.16) p : An elderly woman sitting on a bench.

h : An aged mother sits down.

The systematic procedure to yield two sets of transformed instances under T_{sub} is as follows. We take all the control instances of type *contradiction* that contain antonym word pairs, namely sample I_A. We then apply transformation T_{sub} on premise sentences. This yields a new set of transformed instances, namely I_{TA2}. Then we repeat the procedure but now we apply T_{sub} on hypothesis sentences of I_A, which yields another set of transformed instances, namely I_{TA3} (see Table 5.7.)

Generating Ex Situ Instances We describe the transformation that we use in order to generate both types of *ex situ* samples, namely control and transformed (in

[15] We manually select these from WordNet such that it appears at least $t = 10$ times in the training set on either the premise sentences or the hypothesis sentences (except for 3 words which appear less than 10 but more than 5 times).

Section 5.7.1.4 we describe the purpose of having an *ex situ* condition.) To create control instances we take the *in situ* control instances $i = (p,h) \in I_A \cup I_H$ and transform them via transformation T_{ex}. The transformation is as follows. Given an instance $i = (p,h)$, we randomly select either premise or hypothesis sentence;[16] then, we copy-paste this sentence to form a new sentence pair. At this point, we have an instance where the premise sentence is exactly the same as the hypothesis sentence. After that, we replace w_1 with w_2, or w_2 with w_1, depending on which sentence was selected, in order to preserve the original target word pair (w_1, w_2). We have now generated an *ex situ* control instance $e = (p,h)$ where the target word pair is the same as in the *in situ* control instance counterpart, and the words surrounding the word pair in e are the same in both premise and hypothesis. We do this operation for each *in situ* control instance we have; thus, each *in situ* instance has an *ex situ* instance counterpart.

An example of an *ex situ* control instance is the one in Example 5.17. This instance was obtained from the *in situ* control instance in Example 5.11. In this instance, we can observe that the only difference between premise and hypothesis are the words *sunset* and *sunrise*, which form an antonym word pair.

(5.17) p : A soccer game occurring at sunset.

h : A soccer game occurring at sunrise.

In order to create *ex situ* transformed instances, we apply the same transformation applied to *in situ* instances, namely T_{swap}. Example 5.18 shows the *ex situ* transformed instance generated after transforming the instance in Example 5.17, i.e. after swapping the word pair $(sunset, sunrise)$.

(5.18) p : A soccer game occurring at sunrise.

h : A soccer game occurring at sunset.

5.7.1.4 Experimental Conditions: In Situ and Ex Situ

When we apply transformation T_{swap} to *in situ* control instances, we purposefully alter the instance but we may inadvertently add confounding factors. For example,

[16] We do this in order to control for any possible bias found in either premise or hypothesis sentence since both come from different distributions, i.e. they come from different groups of annotators.

in the transformed instance in Example 5.14, we force the word *bridge* to interact with the expressions *A little girl hugs her brother on a*, on the left, and *in a forest* on the right. But, it may be the case that in no other premise sentence in the training set these interactions occur, i.e. these interactions may be totally new for the systems; thus, this factor, which we call *intra-sentence interaction*, may affect the performance of the system under study, rather than the swap of words per se.

Then, in order to control for the context words surrounding a word pair (w_1, w_2),[17] we create both control and transformed instances where the context words in premise and hypothesis sentences are the same; we call these instances *ex situ*. We now distinguish two experimental conditions, *in situ* and *ex situ*, where the main difference between the two is the form of the instances. In Tables 5.4, 5.5, and 5.6 we show examples of both *in situ* and *ex situ* instances for different class labels. We use the terms *in situ* and *ex situ* to refer to the fact that control and transformed instances are in two forms, in the *average* form of the SNLI data and out of such *average* form; we note that some instances in the SNLI data have a very similar form to our *ex situ* instances, but there are very few of them and thus they are not *average* instances.

	Control	Transformed
In Situ	p : An **elderly** woman sitting on a bench. h : A **young** mother sits down.	p : An elderly woman sitting on a bench. h : An aged mother sits down.

Table 5.4: Examples of control and transformed instances in the *In situ* condition when transformation T_{sub} is used. Label of control instance: *contradiction*; label of transformed instance: *neutral*. In bold text, the word pair where T_{sub} was applied to. No *ex situ* condition exists for this T_{sub}.

	Control	Transformed
In Situ	p : A soccer game occurring at **sunset**. h : A basketball game is occurring at **sunrise**.	p : A soccer game occurring at sunrise. h : A basketball game is occurring at sunset.
Ex Situ	p : A soccer game occurring at **sunset**. h : A soccer game occurring at **sunrise**.	p : A soccer game occurring at sunrise. h : A soccer game occurring at sunset.

Table 5.5: Examples of control and transformed instances in both *In situ* and *ex situ* conditions when transformation T_{swap} is used. Labels of *in situ* control and transformed instances: *contradiction*; labels of *ex situ* control and transformed instances: *contradiction*. In bold text, the word pair where T_{swap} was applied to.

[17] The word pairs are those described in Section 5.7.1.2.

	Control	Transformed
In Situ	p : An **adult** is cutting up onions in a kitchen. h : A **man** is cutting up onions.	p : A man is cutting up onions in a kitchen. h : An adult is cutting up onions.
Ex Situ	p : An **adult** is cutting up onions in a kitchen. h : A **man** is cutting up onions in a kitchen.	p : A man is cutting up onions in a kitchen. h : An adult is cutting up onions in a kitchen.

Table 5.6: Examples of control and transformed instances in both *In situ* and *ex situ* conditions when transformation T_{swap} is used. Labels of *in situ* control and transformed instances: *neutral* and *entailment*, respectively; labels of *ex situ* control and transformed instances: *neutral* and *entailment*, respectively. In bold text, the word pair where T_{swap} was applied to.

5.7.1.5 Factors Controlled

We find several confounding factors to control for during the whole process of creating the test sets to evaluate the systems' robustness. However, only four factors are feasible to be controlled, namely *context words*, *unseen vocabulary*, *average sentence length*, and *order of surrounding words*. In addition, we statistically analyze the confounding factors *insensitivity*, *polarity*, and *unseen word pairs*. These latter three factors are explained in detail in Section 5.7.3.

First, we control for *context words* by setting up an *ex situ* condition, as explained in Section 5.7.1.4. Second, we control for *unseen vocabulary* and *average sentence length* by sampling instances from the training data rather than from development or test data.[18] The first confounding factor refers to introducing words novel to the systems, i.e. words not seen at training time. If we had sampled instances from the SNLI test set in order to create our test sets (as in the first step in Figure 5.1) it is likely that we would have introduced novel words which may affect the performance of the systems since they do not poses any knowledge about the *meaning* of these words. The second confounding factor refers to the number of words in premise and hypothesis sentences. Given that we use the instances sampled from the SNLI data as control instances without any modification, we keep

[18]Controlling for confounding factors by using training instances is a type of control that goes in hand with experimental designs in the behavioral science. When studying factors influencing a subject, it is rather unusual that the environment where the subject is analyzed is different from, or other than, the habitual environment where the subject performs the target activity, since doing so may bring new factors to the study that the experimenter may not be aware of.

unchanged the length of these sentences in each instance.[19] Finally, we control for the *order of surrounding words* in *in situ* instances since the transformation T_{swap} only changes the placement of two words, but it does not change the order of the words surrounding the word pair swapped.

5.7.1.6 Factors Not Controlled

As we mentioned before, there are possible confounding factors which are very difficult, or unfeasible, to control for or to analyze statistically. The first factor we are aware of is *intra-sentence interaction* and arises when we swap a word pair in an *in situ* instance, i.e. when we obtain a transformed instance. This factor refers to the fact that when we apply T_{swap} on a control instance, a word w_2 from the hypothesis sentence replaces a word w_1 from the premise sentence and vice-versa, which leads to the interactions of w_2 with the words in the premise and w_1 with the words in the hypothesis. These interactions may be new to the systems and as such they may influence the systems' behavior due to the lack of any knowledge of such interactions. Controlling for this factor would require to generate transformed samples to study how each single interaction effects on the systems. This study becomes unfeasible since the number of words that we swap combined with the new words they interact with is probably in the order of the hundreds.

A similar confounding factor arises when we generate *ex situ* instances: We may put words in a novel position not seen during training when we copy a premise sentence and paste it as a hypothesis sentence, or the other way around. For example, in Example 5.17 we copied the premise sentence and pasted it as a hypothesis sentence, thus it may be possible that the word *soccer* was never seen in a hypothesis sentence in the training set, and this may affect the performance of the systems (the other words, *game* and *ocurring*, are seen in hypothesis sentences during training.) Another possible confounding factor that we bring in this experimental condition is the *average sentence length*. Having both premise and hypothesis sentences containing the same number of words does not conform to the average SNLI instance,

[19] Furthermore, training data is much more abundant than both development and test data. Sampling instances from either of these two latter sets results in an extremely small sample stopping us from achieving accurate results.

as we show in Table 5.1; in this table we see that the premise sentence is usually longer than the hypothesis sentence in the training set of the SNLI dataset. However, this factor is a by-product of the *ex situ* condition, and controlling for it would mean getting rid of this experimental condition.

A possible way in which we could control for most of the confounding factors is via *laboratory* conditions, where we would aim to be in total control of most of the factors at the cost of abstracting the systems away from their known environment, the SNLI data; however, this would require us to craft instances where all the confounding factors are controlled. Again, we would end up in an unfeasible scenario. In addition, we would sacrifice external validity for internal validity; i.e, we would evaluate the systems on our laboratory instances with some statistical degree of confidence, which does not imply that the results obtained are indicative of the behavior of the systems on SNLI data. Thus, in our case, we have a trade-off between internal and external validity, since we are using original instances from the SNLI dataset, as opposed to using instances hand-crafted under laboratory conditions, at the cost of not control certain factors.

5.7.1.7 Description of Test Sets Used to Evaluate Robustness

We now describe all the test sets used for analyzing both the robustness of the systems and the target factors described in Section 5.7.3.

After applying the procedure described in Sections 5.7.1.2 and 5.7.1.3 we end up with two types of test sets, *in situ* and *ex situ*. Each set contains two types of samples, a control sample, which we denote with the letters I (*in situ*) and E (*ex situ*), and a transformed sample, denoted by I_T and E_T. We recall that the transformed samples are generated from the control samples. We make a further distinction in the samples, namely by the type of word pair that the instances contain. Samples with instances containing antonym word pairs have subscript A, i.e. I_A, I_{TA}, E_A, and E_{TA}; samples with instances containing hypernym or hyponym word pairs are denoted as I_H, I_{TH}, E_H, and E_{TH}. More details about each test set are as follows (a summary can be found in Table 5.7):

I_A : *In situ* sample containing control instances. It only contains *contradiction* instances, and each instance contains an antonym word pair.

I_{TA1} : *In situ* sample containing transformed instances. This sample is generated after applying transformation T_{swap} on I_A. It only contains *contradiction* instances, and each instance contains an antonym word pair.

I_{TA2}, I_{TA3} : *In situ* samples containing transformed instances. These samples are generated after applying transformation T_{sub} on I_A. They may contain instances from any class (*entailment*, *neutral*, *contradiction*); each instance may contain any type of word pair (including synonym word pairs, only used in these cases.)

I_H : *In situ* sample containing control instances. This sample contains all types of instances, though mainly of type *neutral* and *entailment*. Each instance contains either a hypernym or a hyponym word pair.

I_{TH} : *In situ* sample containing transformed instances. It is obtained after applying transformation T_{swap} on I_H. It contains all classes of instances, but mainly *neutral* and *entailment*; each instance contains either a hypernym or a hyponym word pair.

E_A : *Ex situ* sample containing control instances which are generated from I_A. This sample only contains *contradiction* instances, where each instance contains an antonym word pair.

E_{TA} : *Ex situ* sample containing transformed instances. These instances are generated after the application of transformation T_{swap} on E_A. All the instances are class *contradiction*, and each of them contains an antonym word pair.

E_H : *Ex situ* sample containing control instances; these instances are obtained from I_H. This sample only contains instances of class *neutral* and *entailment*, where each instance contains either a hypernym or a hyponym word pair.

E_{TH} : *Ex situ* set containing transformed instances. These instances are generated after the application of transformation T_{swap} on E_H. The instances are either class *entailment* or class *neutral*; each instance contains either a hypernym or a hyponym word pair.

Creation details					Collection details		
Sample	Word Pairs	Type	Transformation	Size	Labels	Labels Changed	Unseen Pairs
I_A	antonym	control		620	*contradiction*		No
I_{TA1}	antonym	transformed	swap	620	*contradiction*	0%	Yes
I_{TA2}	diverse	transformed	substitution	607	all	$\approx 50\%$	Yes
I_{TA3}	diverse	transformed	substitution	608	all	$\approx 50\%$	Yes
E_A	antonym	control	copy-paste	620	*contradiction*		No
E_{TA}	antonym	transformed	swap	620	*contradiction*	0%	Yes
I_H	hyper/hypo	control		648	all		No
I_{TH}	hyper/hypo	transformed	swap	648	all	$\approx 42\%$	Yes
E_H	hyper/hypo	control	copy-paste	644	*neutral*,*entailment*		No
E_{TH}	hyper/hypo	transformed	swap	644	*neutral*,*entailment*	$\approx 93\%$	Yes

Table 5.7: Details of the samples used to test the robustness of the models. *Word Pairs*: Type of word pair contained in the instances of the current sample. *Type*: Type of sample. *Transformation*: Transformation used to obtain the current sample. *Size*: Number of instances in the current sample. *Labels*: class labels found in the current sample. *Labels Changed*: Percentage of instances that have different class label with respect to their control instances counterpart. *Unseen Pairs*: Whether the current sample contains instances with unseen word pairs. Diverse={synonymy, hypernymy, hyponymy}.

5.7.2 Evaluation of Robustness

We aim to measure the generalization abilities of the systems to challenging scenarios, namely the transformed samples we obtained in Section 5.7.1. We envisage two perspectives to analyze the robustness of the systems; the first one by comparing control vs. transformed samples, and the second one by comparing *in situ* vs. *ex situ* scenarios. In all of these comparisons we measure accuracy scores and the difference in such scores.

5.7.2.1 Control vs. Transformed

We evaluate how well the systems generalize to the transformed samples we obtained. In other words, we compare accuracy scores on both control and transform samples to see to what extent our transformations T_{swap} and T_{sub} affected the behavior of the systems. We do this analysis in both experimental conditions, *in situ* and *ex situ*. Our main purpose in this comparison is to find differences in systems' performance between sets of instances where the target word pairs remain in their original position and sets of instances where the target word pairs have been swapped.

Thus, we compare the following samples against each other. First, I_A vs. I_{TA1} and E_A vs. E_{TA}, where we can observe the effect of swapping antonym word pairs in *contradiction* instances in both experimental conditions. Second, I_A vs. I_{TA2} and I_A vs. I_{TA3} where can observe the effect of substituting a word in a word pair in *contradiction* instances. And third, I_H vs. I_{TH} and E_H vs. E_{TH} where we are able to observe the effect of swapping hypernym-hyponym word pairs in all class of instances on both experimental conditions.

5.7.2.2 *In Situ* vs. *Ex Situ*

We also compare the difference in a system's performance between the two experimental conditions, *in situ* and *ex situ*, to figure out whether the words surrounding a swapped word pair influence the systems' behavior. Here we target to measure the difference in score on the following sample pairs. First, we analyze a system's behavior on samples containing only *contradiction* instances, i.e. we analyze behavior on I_A vs. E_A and I_{TA1} vs. E_{TA}, where we can analyze to what extent a system can cope to changes in context words surrounding antonym word pairs. After that, we compare difference in accuracy scores on I_H vs. E_H and I_{TH} vs. E_{TH} to elucidate to what extent the figure to be observed in *contradiction* instances remains in all class of instances containing hypernym-hyponym pairs.

5.7.3 Factors Under Analysis

As part of our experiments, we statistically analyze whether the three confounding factors mentioned in Section 5.1, namely polarity, insensitivity, and unseen word pairs, play a role in the systems' response affecting the accuracy scores that we obtain when we evaluate their robustness. That is, we will elucidate whether the results obtained from the experiments described in Section 5.7.2 are really due to the robustness of the systems or due, to some extent, to the confounding factors. To do so, we use statistical tests, as explained in Section 5.7.4, to unveil any influence of a factor on a system's behavior.

5.7.3.1 Insensitivity

This is an inherent factor to the systems, i.e. an internal factor, rather than an external factor arising in the dataset. This factor relates to the ability of a system to recognize that a control instance was transformed and thus a transformed instance was generated; the way in which we know whether a system recognizes this transformation is via a change of class label between control and transformed instances; i.e., we analyze the response of the systems to both control and transformed instances that have different class labels. For example, we have the control instance shown in Example 5.13, which has associated an *entailment* label, and its transformed counterpart shown in Example 5.14, which has associated a *neutral* label. We feed both instances to a system and we observe if the system recognizes that the transformed instance is different in label from the control instance. We denominate a system *insensitive* if it predicts the same class label for both instances. We test for the influence of this factor via tests of independence between incorrect predictions and change of labels predicted between control and transformed instances.

5.7.3.2 Unseen Word Pairs

This factor may naturally arise after we perform our transformation T_{swap} on control instances, and it indicates that the word pair swapped is a novel word pair (it was unseen at training time.) More concretely, we say that a word pair (w_j, w_i) is unseen if during training time it was not the case that both words, w_i and w_j, were observed co-occurring together in an instance $i = (p,h)$ in the form $w_j \in p$ and $w_i \in h$. This may be the case for some swapped word pairs contained in transformed instances, but it is not the case for un-swapped word pairs in control instances; i.e. all control instances $i = (p,h)$ contain seen word pairs (w_i, w_j) where $w_i \in p$ and $w_j \in h$ are observed co-occurring together in this form at training time.[20]

An example of an unseen word pair is the pair $(sunrise, sunset)$ contained in the transformed instance in Example 5.12; as explained above, it is unseen because there is no instance $i = (p,h)$ on the training set where $sunrise \in p$ and $sunset \in h$. However, it is the case that *sunrise* appears on the premise sentence of an instance

[20]This applies to both experimental conditions, *in situ* and *ex situ*.

$a = (p_a, h_a)$ and *sunset* appears on the hypothesis sentence of another instance $b = (p_b, h_b)$, i.e. $sunrise \in p_a$ and $sunset \in h_b$ but this does not mean that the word pair is seen because the two words did not co-occur in the same instance.

Having unseen word pairs in the transformed instances may affect the performance of a system since there was no learning about the interaction of the two words. Hence, we test whether this factor indeed influences the response of a system by observing if unseen word pairs correlate with incorrect predictions.

5.7.3.3 Polarity

We associate each target word pair with a label –a *polarity*– based on the frequency of the word pair in a class label in training data. We define four classes of polarity where three of them correspond to the classes of the NLI task, namely *entailment*, *neutral*, and *contradiction*.[21] The last polarity corresponds to the label *none* which is designated for those unseen word pairs from Section 5.7.3.2, i.e. pairs not seen in training data but rather on transformed instances. Thus, we *polarize* a word pair with the most frequent label it is seen in training data. For example, we assign the polarity *contradiction* to the word pair $(sunset, sunrise)$ because it is observed mainly in *contradiction* instances, and we assign the polarity *none* to the word pair $(sunrise, sunset)$ because it is an unseen word pair.

Polarity, which acts as a bias (an undesirable pattern), may influence the response of a system. We became aware of this factor by virtue of a manual exploration of the dataset; we observe a certain pattern relating the polarity of word pairs with the class label of the instance containing the pair. For example, it seems that antonym pairs are often correlated with *contradiction* instances, and hypernym pairs with *entailment* instances. Thus, a system may use this bias as cue to predict the class label of the instance. We test the influence of this factor on the systems' response via independence tests between polarity and class label predicted.

[21] To be more precise, we define four more classes, where three of them correspond to combinations of the three NLI class labels, i.e. *entailment-neutral*, *entailment-contradiction*, and *neutral-contradiction*; these labels correspond to cases when a word pair is observed in the same number of instances from such two classes. We call *draw* to the fourth polarity, which indicates that a word pair is seen in the same number of instances from the three classes. We note, however, that these four polarities are rare, i.e. very few word pairs fall in one of these four cases in the training data.

5.7.4 Statistical Analyses

We apply 2-way independence and homogeneity chi-square tests[22] to analyze whether our target factors (Section 5.7.3) play a role on the behavior of the systems under study. These tests allow us to obtain internal validity in our experiments (McDonald, 2014). In all our experiments we use a significance value of $p < 0.0001$ (otherwise stated) in order to correct for any possible false positive rejection of the null hypothesis given that we do multiple tests.[23] In some cases, we use a McNemar test for obtaining a significant comparison of a system's performance between control and transformed samples, where the null hypothesis assumes that the system achieves the same error rate on these two samples (Alpaydin, 2010).[24]

In order to analyze the insensitivity factor, we use transformed instances that have a different class label from that of the control instances they were generated from. Our null hypothesis states an independence between the rate of incorrect predictions and whether the system predicts a different class label for the transformed instances from that predicted for the control instances. In other words, we test whether the system is insensitive to our transformation, and thus predicting the same label for both transformed and controls instances, is a reason for a significant amount of its errors.

When we analyze the transformed samples containing unseen word pairs, our null hypothesis claims an independence between a significant error rate and the presence of such unseen pairs. In other words, we test if the unseen pairs are associated with predicting incorrect class labels.

Finally, in order to analyze if the polarity of word pairs have any influence on a system's behavior, we propose no association between class label predicted and polarity as our null hypothesis; i.e., we test whether a system's predictions correlate with the polarity of the word pairs contained in the instances.

[22] We apply Yate's correction of Fisher exact tests when the sample size is considerably small.
[23] We note that this correction is actually stricter than a Bonferroni correction.
[24] We use the packages StatsModels (Seabold and Perktold, 2010) and SciPy (Oliphant, 2007).

5.8 Experiments and Results

We first compare overall performance between control and transformed samples, i.e. we analyze how robust the systems were to our transformation. Then, we compare overall performance between *in situ* and *ex situ* conditions. After that, we present four experiments, categorized by both the type of experimental condition and target word pair, where in each experiment we analyze in a fine-grained detailed how the factors under analysis played a role in the behavior of the systems.

Before proceeding to the experiments, we give an overview of Table 5.9 which contains the accuracy scores of our experiments. The top-level heading *Whole sample* indicates a system's accuracy score on the control and transformed samples previously described in Section 5.7.1. The next columns contain accuracy scores on subsets of instances taken from such samples. First, *Subset 1* refers to those instances from a transformed sample whose gold labels are different from the gold labels of the instances in the control sample used to generate them; for example, the cell (I_{TA2}, *ESIM*) contains *ESIM*'s accuracy score on the subset of instances from I_{TA2} that have different label from their counterpart instances in I_A. Second, *Subset 2* refers to transformed instances containing unseen word pairs. Finally, *Subset 3* refers to the subset of instances, in either a control or a transformed sample, containing word pairs whose polarity is different from the gold label of the instance; for example, the cell (I_{TA1}, *DAM*) contains the accuracy of *DAM* on those instances from I_{TA1}, a transformed sample with only *contradiction* instances, whose word pairs have polarity other than *contradiction*.

		Insensitivity			Unseen Word Pairs			Polarity		
Exp	sample	ESIM	DAM	CE	ESIM	DAM	CE	ESIM	DAM	CE
1	I_{TA1}				39.33(1)	74.16(1)	19.46(1)	64.4(6)	101.26(6)	30.69(6)
	I_{TA2}	175.34(1)	108.30(1)	73.3(1)						
2	E_A							57.23(8)	53.72(8)	10.78(8)[1]
	E_{TA}				44.72(1)	59.17(1)	15.91(1)	103.47(6)	136.99(6)	34.37(6)
3	I_{TH}	150.92(1)	101.52(1)	90.73(1)	0.178(1)[2]	0.985(1)[3]	0.00(1)[4]	22.72(14)[5]	47.71(14)	25.27(14)[6]
4	E_H							176.38(10)	312.67(10)	261.77(10)
	E_{TH}	252.27(1)	158.62(1)	175.19(1)	0.183(1)[7]	2.43(1)[8]	0.352(1)[9]	105.70(14)	258.09(14)	56.52(14)

Table 5.8: Correlations between the systems' response and confounding factors in terms of χ^2 (chi-square) values. Degrees of freedom are shown next to each correlation value in a parenthesis. All correlations are measured at at p-value of $p < 0.0001$, unless otherwise stated. Other p-values: [1] $p = 0.21$, [2] $p = 0.67$, [3] $p = 0.32$, [4] $p = 0.98$, [5] $p = 0.06$, [6] $p = 0.03$, [7] $p = 0.66$, [8] $p = 0.11$, [9] $p = 0.55$.

		Whole sample			Subset 1: Gold label changes			Subset 2: Unseen word pairs			Subset 3: Polarity ≠ gold label		
Exp	sample	ESIM	DAM	CE	ESIM	DAM	CE	ESIM	DAM	CE	ESIM	DAM	CE
1	I_A	0.970	0.946	0.820							0.900	0.900	0.750
	I_{TA1}	0.933	0.946	0.732				0.600	0.500	0.400	0.681	0.637	0.536
	I_{TA2}	0.721	0.771	0.645	0.554	0.653	0.476						
	I_{TA3}	0.722	0.745	0.646	0.568	0.630	0.535						
2	E_A	0.953	0.958	0.508							0.400	0.500	0.450
	E_{TA}	0.933	0.929	0.480				0.575	0.500	0.175	0.565	0.492	0.260
3	I_H	0.898	0.819	0.828							0.836	0.701	0.733
	I_{TH}	0.648	0.691	0.543	0.315	0.509	0.271	0.694	0.777	0.555	0.719	0.697	0.586
4	E_H	0.771	0.849	0.742							0.715	0.707	0.461
	E_{TH}	0.576	0.788	0.534	0.551	0.783	0.516	0.527	0.666	0.472	0.631	0.674	0.507

Table 5.9: Accuracy scores of all systems. *Exp*: experiment number. *Whole sample*: accuracy scores on the whole sample indicated by the second column, namely *sample*. *Subset 1*: subset of instances from the whole transformed sample that have different label with respect to the control instances they were generated from. *Subset 2*: subset of transformed instances that contain word pairs unseen at training time. *Subset 3*: subset of control or transformed instances containing word pairs whose polarity does not match the instance's gold label.

5.8.1 Evaluation of Robustness

5.8.1.1 Control vs. Transformed

We start our exploration of results with those where the systems seem to be robust when control instances are compared to their transformed instances counterpart. The *ESIM* system shows robustness to our transformation T_{swap}, as far as accuracy scores can show, when the instances are *contradiction* and they contain antonym word pairs; this seemingly robust behavior applies in both conditions, *in situ* and *ex situ*. Table 5.9 shows that the difference in accuracy between a control sample and a transformed sample is small: 0.037 points dropped between I_A and I_{TA1}, and 0.02 points dropped between E_A and E_{TA}. However, in both cases the difference in scores, though small, may be significant according to a McNemar test: *in situ*: $\chi^2(1) = 11.80, p < 0.001$, *ex situ*: $\chi^2(1) = 10.08, p < 0.01$. These two cases are the only ones where *ESIM* seems to be robust.

Similarly, the *DAM* system seems to be robust in the same cases as *ESIM*. The drop in score between I_A and I_{TA1} is zero, and a McNemar test finds no evidence to support a significant difference in error rates; *DAM* seems to be totally robust in this case, though the reason for this result may not be precisely robustness but rather

a confounding factor. The difference in the *ex situ* scenario (E_A and E_{TA}) is 0.029 points, a very small drop in performance; however, a McNemar test ($\chi^2(1) = 16.05$, $p < 0.0001$) shows strong evidence for the system having significantly worse error rate in the transformed sample compared to that in the control sample. These are the only two cases in which the *DAM* system seems to be robust to transformation T_{swap}.

Finally, the *CE* system shows a robust behavior only when the control sample E_A is compared with the transformed sample E_{TA}. The drop in performance is only 0.028 points; however, this result may not be really indicative of a robust behavior, since the accuracy score on the control sample is already very low (0.508 points).

As we can see, *ESIM* and *DAM* seem to be robust when the control instances are of class *contradiction*, they contain an antonym pair, and the transformation is a swap of words, namely transformation T_{swap}. However, this behavior is not persistent if the transformation is a substitution, namely transformation T_{sub}. We recall that this transformation is only used to analyze the insensitivity of the systems (Section 5.8.2.1), but in Table 5.9 we can see that the three systems drop accuracy when we apply this transformation on control instances (I_A vs. I_{TA2} and I_A vs. I_{TA3}), possibly because this transformation changes the gold label of the transformed instances with respect to the control instances. More concretely, *ESIM* drops up to 0.249 points, around 25% of its accuracy on the control sample, while *DAM* drops up to 0.201 points, and *CE* loses up to 0.175 points.

On the other hand, the systems exhibit a non-robust behavior to transformation T_{swap} when the control instances are of class *entailment* or *neutral*. In the *in situ* condition (I_H vs. I_{TH}), drops in accuracy are 0.250, 0.128, and 0.285 points for *ESIM*, *DAM*, and *CE*, respectively. In the *ex situ* condition, drops in score between control (E_H) and transformed (E_{TH}) samples are 0.195, 0.061, and 0.208 for *ESIM*, *DAM*, and *CE*, respectively. We observe that in both conditions *ESIM* and *CE* have the biggest drops in performance, being quite similar the drops between the two systems; in what could be seen as a moderate drop from *DAM* in the *ex situ* condition is, nonetheless, a significant drop according to a McNemar test

($\chi^2(1) = 7.25, p < 0.01$).

5.8.1.2 *In Situ* vs. *Ex Situ*

We now report accuracy differences between the two experimental conditions. In the case of *contradiction* instances containing antonym word pairs, *ESIM* and *DAM* seem to be not affected by the type of context words surrounding an antonym pair; the differences in score between I_A and E_A are minimal (0.017 points for *ESIM* and 0.012 points for *DAM*), as well as those between I_{TA1} and E_{TA} (0 points for *ESIM* and 0.017 points for *DAM*). In dissonance with these results, the *CE* system is heavily affected when the context is repeated in premise and hypothesis sentences with drops in score of 0.312 and 0.252 points when the results on the *in situ* samples I_A and I_{TA1} are compared with those of the *ex situ* condition, E_A and E_{TA}, respectively.

However, the picture portrayed for *contradiction* instances where *ESIM* and *DAM* seemed to be robust to the context type, is no longer preserved for *entailment* and *neutral* instances. In the new picture, *ESIM* shows difficulty to cope with the *ex situ* condition when compared to its *in situ* counterpart, similar to the picture portrayed by *CE*, while *DAM* shows the opposite behavior; *DAM* seems to cope better with *ex situ* instances rather than with *in situ* ones. Drops in accuracy for *ESIM* go up to 0.127 points while this figure is up to 0.086 points for *CE*. *DAM* reaches its peak in performance when tested on the E_H sample with 0.03 improvement with respect to I_H.

5.8.2 Influence of Target Factors

We saw in Section 5.8.1.1 that the systems seem to be robust in certain cases and non-robust in other cases. However, it is unclear to what extent this behavior is the result of the systems' abilities, such as linguistic abilities learned, or rather due to the influence of one of the confounding factors under study. In this section we provide experiments to study which of our target factors –insensitivity, unseen word pairs, and polarity– have an influence on the response of the systems (see Section 5.7.3 for a description of these factors.)

5.8.2.1 Experiment 1: Swapping Antonym Word Pairs In *In Situ* Instances

ESIM and *DAM* seem to be robust to swapping words in an antonym word pair when this pair is placed in *contradiction* instances, regardless of the experimental condition, as we described earlier. However, we show here that this behavior is in part due to the insensitivity of the systems to changes in the control instances, and also due the polarity of the word pairs under analysis.

Insensitivity To study this factor, we use the control sample I_A and the transformed samples I_{TA2} and I_{TA3}. Our first piece of evidence towards unmasking the effect of this factor are accuracy scores in the column Subset 1 from Table 5.9. We observe that all systems' accuracy on transformed instances whose gold label is different from that of their control instances counterpart are much lower than accuracy scores on the complete transformed sample. An even bigger gap in score can be seen when accuracy scores of Subset 1 from I_{TA2} and I_{TA3} are compared against accuracy scores of the control sample I_A. The three systems have a poor performance on instances from Subset 1, and the accuracy of the state-of-the-art system, *ESIM*, is very close to that of our most simple system, *CE*. However, accuracy scores alone may not be really indicative, therefore we confirm the influence of insensitivity via chi-square tests.

In Tables 5.10, 5.11, 5.12 we see the contingency tables used for the 2-way independence tests; these tables comprise those instances in Subset 1 extracted from I_{TA2} that have different class label from the control instances in I_A. For all systems, we find very strong evidence towards incorrect predictions being mainly due to the systems predicting the same labels for both transformed and control instances (*ESIM*: $\chi^2(1) = 175.34$, *DAM*: $\chi^2(1) = 108.30$, *CE*: $\chi^2(1) = 73.33$, $p < 0.0001$) (see Table 5.8). In other words, insensitivity, the prediction of same labels on transformed instances as on control instances, is the main source of error in the three systems.

A different picture would be portrayed if the main source of error was, for example, the prediction of an incorrect label for a transformed instance that is dif-

ferent from the label of the control instance; take as an example a system predicting the label *neutral* for a transformed instance whose gold label is *entailment*, and let's suppose that the control instance from which the transformed instance was generated has gold label *contradiction*. The system would still make a mistake but it would not be deemed as insensitive.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	155	31
no change	8	100

Table 5.10: Contingency table for *ESIM* (Experiment 1): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	179	37
no change	13	65

Table 5.11: Contingency table for *DAM* (Experiment 1): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	95	29
no change	45	125

Table 5.12: Contingency table for *CE* (Experiment 1): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

Unseen Word Pairs Handling antonym pairs in a new order seems to be problematic for the three systems. In the subset of transformed instances that contain unseen antonym pairs, namely Subset 2 of I_{TA1} (Table 5.9), we observe that the systems get accuracy scores much lower than the scores on the complete sample; concretely, the

drops in score are 0.333, 0.446, 0.332 for *ESIM*, *DAM*, and *CE*, respectively. These results are strongly backed up by homogeneity tests as shown in Table 5.8 (*ESIM*: $\chi^2(1) = 39.33$, *DAM*: $\chi^2(1) = 74.16$, *CE*: $\chi^2(1) = 19.46$, $p < 0.0001$), which are computed from Tables 5.13, 5.14, 5.15. We note, however, that the number of transformed instances containing unseen antonym pairs is very small, 40 instances; thus, these results are to be taken with some precaution.

	Word pairs	
Predictions	seen	unseen
correct	555	24
incorrect	25	16

Table 5.13: Contingency table for *ESIM* (Experiment 1): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	567	20
incorrect	13	20

Table 5.14: Contingency table for *DAM* (Experiment 1): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	438	16
incorrect	142	24

Table 5.15: Contingency table for *CE* (Experiment 1): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

Polarity We use sample I_{TA1} for this experiment, where we test an association between the polarity of our target word pairs and the class labels predicted by the systems. We note that in this transformed sample only 11% of the word pairs have a polarity different from *contradiction*; this means that a system predicting class labels based only on this factor would achieve an accuracy score of 0.89.

Comparing Subset 3 of I_A with Subset 3 of I_{TA1} (Table 5.9), where the polarity of a word pair is different from the gold label of the instance containing it, we observe that *ESIM*, *DAM*, and *CE* drop 0.219, 0.263, and 0.214 accuracy points, respectively. The systems probably memorized the instances from the control sample, and thus they did not need to use polarity, but for predicting class labels in the transformed sample they seem to rely on this factor. According to our independence tests, computed from Tables 5.16, 5.17, 5.18,[25] we clearly reject the null hypothesis and confirm a correlation between polarity of word pairs and class label predicted (*ESIM*: $\chi^2(6) = 64.40$, *DAM*: $\chi^2(6) = 101.26$, *CE*: $\chi^2(6) = 30.69$, $p < 0.0001$).

As a point of comparison with a perfect classifier that achieves accuracy score of 1.0 on I_{TA1}, the statistic obtained would not reject the null hypothesis, and would imply no association between polarity and class label predicted.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	6	20	0
Contradiction	7	532	12
Entailment	0	3	0
None	9	24	7

Table 5.16: Excerpt of contingency table for *ESIM* (Experiment 1): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	5	21	0
Contradiction	5	543	3
Entailment	0	3	0
None	8	20	12

Table 5.17: Excerpt of contingency table for *DAM* (Experiment 1): Predictions of class labels distributed according to the word pair polarity they match with.

[25] We leave out of the tables polarity classes *entailment-neutral*, *entailment-contradiction*, *neutral-contradiction*, and *draw*, since very few instances fall in these categories.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	5	18	3
Contradiction	81	417	53
Entailment	0	3	0
None	21	16	3

Table 5.18: Excerpt of contingency table for *CE* (Experiment 1): Predictions of class labels distributed according to the word pair polarity they match with.

5.8.2.2 Experiment 2: Swapping Antonym Word Pairs In *Ex Situ* Instances

In this experiment, we study whether the target factors influence the systems' behavior when fed with *ex situ* instances as they are influenced when fed with *in situ* instances. Though, we skip the study on insensitivity since this requires to generate a new sample via transformation T_{sub}, which may add an extra-layer of alterations to the *ex situ* control sample yielding results that may be difficult to interpret.

Unseen Word Pairs We find the same problem as in the *in situ* condition on those transformed *ex situ* instances that contain word pairs in an unseen order. Comparing accuracy scores of the whole sample E_{TA} with this subset of instances (Subset 2 in Table 5.9), we can see, again, big drops in accuracy of at least 0.305 points (*CE*) and up to 0.429 points (*DAM*). Our homogeneity tests (Tables 5.19, 5.20, and 5.21) show strong evidence for these drops in accuracy being associated with the unseen word pairs present in the transformed instances (*ESIM*: $\chi^2(1) = 44.72$, *DAM*: $\chi^2(1) = 59.17$, *CE*: $\chi^2(1) = 15.91$, $p < 0.0001$) as show in Table 5.8.

We compare accuracy scores of transformed instances in Subset 2 from both experimental conditions, I_{TA1} and E_{TA}; we observe that *ESIM* and *DAM* seem to be robust to the context type, and thus drops in accuracy seem to be explained by the unseen word pairs. On the other hand, the extremely low score of *CE* seem to indicate that it is affected by both, the context type and the unseen pairs.

Polarity From the accuracy scores on the instances in Subset 3 from both samples control (E_A) and transformed (E_{TA}), we can see that the three systems have severe

	Word pairs	
Predictions	seen	unseen
correct	556	23
incorrect	24	17

Table 5.19: Contingency table for *ESIM* (Experiment 2): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	556	20
incorrect	24	20

Table 5.20: Contingency table for *DAM* (Experiment 2): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	291	7
incorrect	289	33

Table 5.21: Contingency table for *CE* (Experiment 2): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

trouble when the instances' gold labels mismatch the polarity of the word pairs they contain, except for *CE* where the drop in accuracy in Subset 3 of E_A with respect to the whole sample is moderate (0.058). However, in the same scenario (Subset 3 of E_A vs. whole E_A), *ESIM* and *DAM* drop scores by 0.553 and 0.458 points, respectively. Our tests of independence confirm the influence of word pairs polarities on the behavior of both *ESIM* ($\chi^2(8) = 57.23,\ p < 0.0001$) and *DAM* ($\chi^2(8) = 53.72,\ p < 0.0001$), but the tests do not reject the null hypothesis in the case of *CE* ($\chi^2(8) = 10.78,\ p = 0.21$), meaning that we have no evidence to support the claim that this factor influences *CE*'s behavior when fed with the sample E_A.

However, when the *CE* system faces the transformed sample, E_{TA}, it significantly drops accuracy by 0.22 points on those instances mismatching polarity and

gold label, namely Subset 3 of E_{TA} ($\chi^2(6) = 34.37$, $p < 0.0001$); we find a similar behavior shown by both *ESIM*, with a significant drop in performance of 0.368 points ($\chi^2(6) = 103.47$, $p < 0.0001$), and *DAM*, with also a significant drop of 0.437 points ($\chi^2(6) = 136.99$, $p < 0.0001$) (see Tables 5.22, 5.23, and 5.24.)

Comparing the scores on Subset 3 of the *ex situ* condition with those of Subset 3 in the *in situ* condition, we see a degradation of performance in the former case. It seems that the three systems, besides relying on the polarity bias, they relayed on the context type, being the context words in the *in situ* condition a more familiar type of context for the systems, because such context words were observed at training time.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	0	14	12
Contradiction	0	540	11
Entailment	0	2	1
None	2	23	15

Table 5.22: Excerpt of contingency table for *ESIM* (Experiment 2): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	2	11	13
Contradiction	0	542	9
Entailment	0	3	0
None	1	20	19

Table 5.23: Excerpt of contingency table for *DAM* (Experiment 2): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	11	11	4
Contradiction	135	280	136
Entailment	0	0	3
None	23	7	10

Table 5.24: Excerpt of contingency table for *CE* (Experiment 2): Predictions of class labels distributed according to the word pair polarity they match with.

5.8.2.3 Experiment 3: Swapping Hypernym and Hyponym Word Pairs In *In Situ* Instances

We saw in Section 5.8.1.1 that the three systems dropped accuracy, by a wide margin, on the transformed sample I_{TH} with respect to its control sample counterpart. We investigate whether our target factors are playing a role in such a behavior.

Insensitivity Looking at the subset of transformed instances that have different class label from those in the control sample (Subset 1 of I_{TH} in Table 5.9), we observe a further drop in accuracy score across the systems. Comparing accuracy on the whole transformed sample against this subset of instances, we observe the following drops: 0.333 points for *ESIM*, 0.182 points for *DAM*, and 0.272 points for *CE*. We find strong evidence for these results to be associated with the insensitivity of the systems to the difference between control and transformed instances, namely the position of the word pairs swapped; this strong evidence is given by our independence tests (*ESIM*: $\chi^2(1) = 150.92$, *DAM*: $\chi^2(1) = 101.52$, *CE*: $\chi^2(1) = 90.73$, $p < 0.0001$) as shown in Table 5.8. These tests are computed based on Tables 5.25, 5.26, and 5.27.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	70	14
no change	16	173

Table 5.25: Contingency table for *ESIM* (Experiment 3): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	115	31
no change	24	103

Table 5.26: Contingency table for *DAM* (Experiment 3): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	59	34
no change	15	165

Table 5.27: Contingency table for *CE* (Experiment 3): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

Unseen Word Pairs We note that accuracy scores on Subset 2 of I_{TH}, which only contains instances that have unseen word pairs, are better for all systems than the accuracy scores on the complete transformed sample I_{TH}. It seems that the systems are robust to unseen hypernym and hyponym word pairs, as opposed to when they faced unseen antonym pairs. As a confirmation of this finding, our homogeneity tests find no correlation between the presence of unseen pairs with incorrect predictions (*ESIM*: $\chi^2(1) = 0.178,\ p = 0.67$, *DAM*: $\chi^2(1) = 0.985,\ p = 0.32$, *CE*: $\chi^2(1) = 0.00036, p = 0.98$) as shown in Table 5.8; see also Tables 5.28, 5.29, 5.30 for a view of the contingency tables used for computing the tests.

	Word pairs	
Predictions	seen	unseen
correct	395	25
incorrect	217	11

Table 5.28: Contingency table for *ESIM* (Experiment 3): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	420	28
incorrect	192	8

Table 5.29: Contingency table for *DAM* (Experiment 3): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	332	20
incorrect	280	16

Table 5.30: Contingency table for *CE* (Experiment 3): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

Polarity Looking at the accuracy results of Subset 3 of the control sample I_H (Table 5.9), we see that the three systems lose accuracy when the polarity of the word pairs differ from the gold label of the instances they are contained in. Independence tests strongly confirm this association, namely between systems' error rate and word pairs polarities (*ESIM*: $\chi^2(10) = 157.76$, *DAM*: $\chi^2(10) = 182.76$, *CE*: $\chi^2(10) = 168.40$, $p < 0.0001$) (see Tables 5.31, 5.32, and 5.33). However, this figure is only maintained by *DAM* on the transformed sample ($\chi^2(14) = 47.71$, $p < 0.0001$); *ESIM* and *CE* seem to cease using polarity as a cue for predicting class labels when faced with sample I_{TH}, according to our independence tests (*ESIM*: $\chi^2(14) = 22.72$, $p = 0.06$, *CE*: $\chi^2(14) = 25.27$, $p = 0.03$), which show weak evidence for an association between polarity and prediction of labels.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	126	60	36
Contradiction	27	21	15
Entailment	67	54	221

Table 5.31: Excerpt of contingency table for *ESIM* (Experiment 3): Predictions of class labels distributed according to the word pair polarity they match with.

5.8.2.4 Experiment 4: Swapping Hypernym and Hyponym Word Pairs In *Ex Situ* Instances

In Section 5.8.1.1, we saw big drops in accuracy scores, between control and transformed samples, from *ESIM* and *CE*, and a moderate, but significant, drop from *DAM*. We now analyze the influence of our target factors into these results.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	132	57	33
Contradiction	30	19	14
Entailment	64	53	225

Table 5.32: Excerpt of contingency table for *DAM* (Experiment 3): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	119	68	35
Contradiction	23	20	20
Entailment	56	59	227

Table 5.33: Excerpt of contingency table for *CE* (Experiment 3): Predictions of class labels distributed according to the word pair polarity they match with.

Insensitivity Accuracy scores on Subset 1 of E_{TH} are slightly lower than the scores for the complete sample, across all systems. Thus, it is not very clear whether insensitivity has an impact on the systems' behavior. However, independence tests (see Tables 5.34, 5.35, and 5.36) show very strong evidence in favour of such an impact (*ESIM*: $\chi^2(1) = 252.27$, *DAM*: $\chi^2(1) = 158.62$, *CE*: $\chi^2(1) = 175.19$, $p < 0.0001$) as shown in Table 5.8. Comparing the accuracy scores on Subset 1 of E_{TH} with those on the same subset of the *in situ* condition, namely Subset 1 of I_{TH}, we see that the three systems seem to be more insensitive to instances closer to the training set; i.e., the three systems have lower accuracy on the Subset 1 of the *in situ* than on Subset 1 of the *ex situ* transformed instances.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	233	24
no change	98	245

Table 5.34: Contingency table for *ESIM* (Experiment 4): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	409	38
no change	61	92

Table 5.35: Contingency table for *DAM* (Experiment 4): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

	Distribution of predictions	
Labels predicted	correct	incorrect
change	215	49
no change	95	241

Table 5.36: Contingency table for *CE* (Experiment 4): Predictions of class labels distributed according to matching or not the gold labels. Only transformed instances that have gold labels different from those of their control instances counterpart are used.

Unseen Word Pairs Accuracy scores on Subset 2 of E_{TH}, when compared to the scores on the whole sample, seem to indicate that all systems struggle with instances containing unseen word pairs. However, homogeneity tests, based on Tables 5.37, 5.38, and 5.39, find either no evidence or weak evidence for such an indication (*ESIM*: $\chi^2(1) = 0.183,\ p = 0.66$, *DAM*: $\chi^2(1) = 2.43,\ p = 0.11$, *CE*: $\chi^2(1) = 0.352,\ p = 0.55$). This evidence, together with that from Experiment 3 in the *in situ* condition, works towards the hypothesis that this class of systems are robust to unseen hypernym and hyponym pairs.

	Word pairs	
Predictions	seen	unseen
correct	352	19
incorrect	256	17

Table 5.37: Contingency table for *ESIM* (Experiment 4): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	484	24
incorrect	124	12

Table 5.38: Contingency table for *DAM* (Experiment 4): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

	Word pairs	
Predictions	seen	unseen
correct	327	17
incorrect	281	19

Table 5.39: Contingency table for *CE* (Experiment 4): Predictions of class labels distributed according to whether they contain a seen or an unseen antonym word pair.

Polarity When we compare accuracy scores on Subset 3 with the scores on the complete sample, for both control and transformed samples, we observe a tendency across all systems, namely a drop in performance; i.e., the systems lose accuracy when the polarities of the word pairs mismatch the gold labels of the instances. This tendency is strongly backed up by independence tests in both the control sample (*ESIM*: $\chi^2(10) = 176.38$, *DAM*: $\chi^2(10) = 312.67$, *CE*: $\chi^2(10) = 261.77$, $p < 0.0001$) and the transformed sample (*ESIM*: $\chi^2(14) = 105.70$, *DAM*: $\chi^2(14) = 258.09$, *CE*: $\chi^2(14) = 56.52$, $p < 0.0001$) as shown in Table 5.8. Tables 5.40, 5.41, and 5.42 show an excerpt of the contingency tables used for computing the statistics for the transformed sample.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	83	26	195
Contradiction	30	12	46
Entailment	6	15	172
None	10	1	25

Table 5.40: Excerpt of contingency table for *ESIM* (Experiment 4): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	211	9	84
Contradiction	25	16	47
Entailment	13	16	164
None	14	4	18

Table 5.41: Excerpt of contingency table for *DAM* (Experiment 4): Predictions of class labels distributed according to the word pair polarity they match with.

Polarity \ Prediction	Neutral	Contradiction	Entailment
Neutral	116	19	169
Contradiction	29	12	47
Entailment	27	10	156
None	10	4	22

Table 5.42: Excerpt of contingency table for *CE* (Experiment 4): Predictions of class labels distributed according to the word pair polarity they match with.

5.9 Discussions and Conclusions

We start this section by answering our research questions and then we elaborate on each of them and on the results obtained from the experiments done.

We found that none of the systems under study is robust on all of our transformed samples when accuracy scores on these samples are compared to accuracy scores on the SNLI data. In some cases, *ESIM* and *DAM* achieve high scores on our samples, but we show via statistical tests that part of the reason for these scores is due to the influence of confounding factors. We found that the way in which the systems are affected by these confounding factors is very similar; in most of the cases, the three systems use polarity of word pairs to predict class labels and in other cases they are benefited by being insensitive to transformations in the instances. In addition, the systems are affected when instances contain unseen antonym word pairs, but surprisingly, they are not affected by unseen hypernym-hyponym pairs. We believe that the three systems are able to cope with unseen hypernym and hyponym word pairs because they may have learned some characteristics of hypernymy, such as generality and similarity (we will elaborate on this two features on

Chapter 6); for example, if a system sees at training time entailment instances where hyponym word pairs appear, such as $(cat, animal)$, then it will learn for this class of instances that the more abstract concept (in this case *animal*) will always appear in the hypothesis sentence, while the less abstract concept (*cat* in our example) will appear on the premise sentence; and similarly with other pairs such as $(dog, animal)$, $(car, vehicle)$, $(bicycle, vehicle)$, and so on; thus, the systems probably learned that one concept is more *general* than the other. The systems probably also learned that both concepts are *similar* because they appear in similar contexts; for example, *cat* and *animal* will often appear with words such as *food*, *prey*, *hunt*, *eat*, and so on. Furthermore, hypernym pairs, such as $(animal, cat)$, tend to appear in neutral instances, providing similar information as the one we just described. Therefore, when the systems are presented with a new hyponym pair, say $(dog, animal)$, they may infer that this is a hyponym pair because *animal* is a concept with the characteristics of being abstract (general) across instances and being similar to *dog*. And similarly for the hypernym pair $(animal, dog)$.

5.9.1 On Systems' Robustness on Transformed Instances Containing Antonym Word Pairs

Our chi-square tests provide evidence for the targets factors playing a role in the response of the systems. Thus, these statistical tests show that some of the reasons for *ESIM* and *DAM* showing an apparently robust behavior, and for *CE* not dropping accuracy dramatically, are two out of three of the target factors. On the one hand, the three systems seem to be insensitive to small changes, such as those made by transformation T_{sub} which is a simple substitution of a word in either the premise or hypothesis sentence. This insensitivity seems to be a possible explanation of why these systems achieved a very good accuracy on the transformed sample I_{TA1}. The systems' insensitivity to the difference between the control and transformed samples (I_A and I_{TA1}) helped them to predict the same labels for both samples, namely *contradiction*, achieving a very good score on the transformed sample because the gold labels of these instances turned out to be the same as the gold labels of the control instances. This, in turn, also explains why the two systems have a big gap

in performance between sample I_{TA1} and those transformed samples that have different gold labels from the control sample, namely I_{TA2} and I_{TA3}.

Another possible reason for *ESIM*'s and *DAM*'s good scores, though unrelated to what we call *robust behavior*, and for *CE* not losing more accuracy points, is the polarity of word pairs. The three systems learned to correlate the most frequent class a word pair is seen at training time with the label of the instance it is contained in; this correlation influenced the systems' response each time a polarized word pair was contained in an instance. In simple words, the systems learned that if a word pair (w_1, w_2) happens mostly in *contradiction* instances, then when they are fed with a test instance containing this word pair it is likely that such an instance will be of class *contradiction* as well, and thus the systems predict such a class label perhaps without reading the surrounding words. The systems learned to do so due to the design of the SNLI dataset. In the test sets we created, we can observe that annotators tend to use antonyms in *contradiction* instances much more often than on *entailment* or *neutral* instances. And similarly with hypernym-hyponym pairs with the latter two classes. In this way, *ESIM*, *DAM*, and *CE* learned to predict *contradiction* class based on whether an instance contained a polarized antonym pair. And similarly for *entailment* and *neutral* classes containing hypernym or hyponym pairs.

On the other hand, the presence of unseen antonym pairs affected the performance of the three systems, though the number of instances containing such pairs is very low (around 7%); thus the overall performance on the transformed sample did not suffer such a big drop. Nevertheless, on this small sample our independence tests show strong evidence for *ESIM*, *DAM*, and *CE* not having learned a symmetric antonym relation. Though our statistical tests suggest that the systems did not learn antonymy, this evidence, however, does not imply that the systems are not able to learn this semantic relation from data. It may be possible that with more instances, or with a different design of the data, the systems are able to do so.

Thus, overall, insensitivity and polarity seem to have contributed to the seemingly robust performance of *ESIM* and *DAM* on *contradiction* instances in the trans-

formed sample I_{TA1}, with a limited number of unseen antonym pairs not deeply affecting its overall score. Therefore, we conclude that both *ESIM* and *DAM* are not as robust as its overall accuracy score implies on any of the transformed *in situ* samples, namely I_{TA1}, I_{TA2}, and I_{TA3}.

Analyzing the same class of instances, *contradiction*, but on the *ex situ* condition, we find very similar results to those in the *in situ* condition. The three systems, again, use polarity as a cue to predict the class label of an instance. And their performance is hurt by unseen antonym pairs, though the number of instances containing such a type of pair is again 7%, thus any damage to score is tiny. Overall, polarity seems to be a plausible explanation for why *ESIM* and *DAM* are robust in both cases, the *ex situ* control condition (E_A) and the challenging sample (E_{TA}), as our statistical tests suggest. The case of *CE* is a bit different from the cases of the other two systems; *CE* gets poor scores on both samples. A possible explanation for such a low score on the sample E_A may be because it does not use polarity as a cue for predicting class labels, as the other two systems do. However, *CE* seems to use polarity for predicting labels on the transformed sample, as both our statistical tests show and as we can see on the score of Subset 3 from E_{TA} where it performs even poorer on instances where polarity cannot be used.

However, we are aware that other possible factors, such as confounding factors or systems' abilities learned, may also play a role in their behavior.

5.9.2 On Systems' Robustness on Transformed Instances Containing Hypernym-Hyponym Word Pairs

A clearer picture of the systems' non-robust behavior can be seen when the types of instances are *entailment* or *neutral* and they contain hypernym or hyponym word pairs. This behavior is clearer than with *contradiction* instances containing antonyms pairs since the difference between control and transformed samples is exposed: 42% of transformed *in situ* instances change label with respect to their control instances counterpart, while the same figure is observed in 93% of the transformed *ex situ* instances. In both experimental conditions, the three systems see their performance significantly affected by their insensitivity to the small change

generated by our transformation, specially in the *in situ* condition where we see big drops in accuracy on the subset of transformed instances that have different label from the control instances, being *ESIM* the most affected with 0.333 points lost.

If insensitivity was the only factor affecting the systems' behavior then it would be plausible to think that the biggest drops in score from the control sample to the transformed sample should be seen in the *ex situ* condition, since changes in labels occurs in almost all the transformed sample (with respect to the control sample). However, it is the other way around; the biggest drops in score, for the three systems, are seen in the *in situ* condition, i.e. the differences in score between I_H and I_{TH} is bigger than the difference between E_H and E_{TH}. A possible explanation for this phenomenon is the influence of polarity on the systems' behavior. In the *in situ* condition, we find weak evidence for *ESIM* and *CE* using polarities of word pairs to predict labels of the transformed instances. In contrast, in the *ex situ* condition there is strong evidence for the systems using polarity. However, *DAM* seems to use polarity in both conditions, *in situ* and *ex situ*; thus, it may be the case that this system's behavior is affected by the form of the *ex situ* instances. Overall, it seems that polarity of word pairs may be a crucial factor for the performance of the three systems.

In contrast to the case of *contradiction* instances containing antonym word pairs where the three systems failed to learn antonymy, the systems seem to have learned an asymmetric hypernymy relationship. Our statistical tests show strong evidence for the systems to correctly responding to unseen hypernym-hyponym pairs, a clear proof of robustness. We can support this finding with our main finding in Chapter 6; it seems that ReLe systems are able to learn hypernymy.

5.9.3 On the Accuracy on Transformed Instances vs. Accuracy on SNLI Development Set

Up to this point we have contrasted a control sample with a transformed sample for a combination of experimental conditions and classes of instances. Across these studies, we have seen that accuracy scores showed a misleading picture of *ESIM* and *DAM* on transformed instances of class *contradiction*; it seemed that both systems

had properly labelled these instances. However, we provided statistical evidence against a robust behavior and in favour of confounding factors playing a role in the systems' behavior. Nevertheless, accuracy scores provided a signal for a clear impairment of not only *ESIM* and *DAM* but also *CE* on transformed instances of classes *entailment* and *neutral*, regardless of the experimental condition. Our statistical tests confirmed such an impairment by providing evidence for our target factors having an effect in the systems' behavior. Nonetheless, we saw evidence of robust behavior when the three systems faced unseen hypernym-hyponym pairs.

However, these analyses may be disembodied from a wider context: How do these drops in performance compare with the overall performance of the system?

The accuracy scores of *ESIM*, *DAM*, and *CE* on the development set of the SNLI dataset are 0.882, 0.854, and 0.782 points, respectively. Comparing these scores to the scores obtained for any transformed *in situ* sample, we find only one case where *ESIM* and *DAM* seem to be robust exceeding any expectation, namely on the transformed sample I_{TA1}. Nevertheless, we already explained how the scores on this sample are deceptive and we showed that *ESIM* and *DAM* actually fail to be robust on this sample. In the case of *CE*, there is only one case where it does not drop accuracy in a drastic way, i.e. on I_{TA1}, though reasons for this behavior are the same as the reasons for the apparent robust behavior of the two other systems. On the other hand, if we look at the *ex situ* samples, we find one case where *ESIM* and *DAM*, again, seem to exceed expectations with respect to their overall performance on the SNLI development set, namely on sample E_{TA} with scores of 0.933 and 0.929 points, respectively. However, we can attribute, to some extent, the exploitation of the polarities of word pairs to their strong performance; thus leaving unclear to what extent the accuracy results observed really are due to robust behavior. Thus, we conclude that, in general, *ESIM*'s and *DAM*'s behavior in our test sets mismatch their state-of-the-art performance on the SNLI development set by failing to maintain their robustness.

5.9.4 On Common Behavioral Patterns Across the Systems

A natural follow-up question to our previous conclusion is that of comparing the behavioral patterns we observed from the three systems on our test sets: Are there any common behavioral patterns among these ReLe systems? According to our experiments, and as we have seen through out this section of discussions, the former state-of-the-art system, *DAM*, behaves in most cases similarly to *ESIM*; it is insensitive, it uses polarity as a cue for predicting labels, it seemed to have not learn antonymy, though it seemed to have learned hypernymy. All of these behavioral patterns are found at both systems at par in almost all the test samples. Thus robustness of these two systems are comparable. However, there are two cases where *DAM* shows better performance than *ESIM*. The first one is on the sample E_H, where *DAM* loses minimal accuracy with respect to its accuracy on the development set of the SNLI dataset; on this sample, *ESIM* is not near to match its own accuracy score on the development set. The second case is when we compare scores on the control samples I_H and E_H; *DAM* is not affected by the context words surrounding hypernym or hyponym word pairs and achieves similar results on both samples, whereas *ESIM* is affected by such a factor. Another comparison we provide is against a simpler version of *ESIM*, namely *CE*. Similar to our observations from the previous comparison, the same behavioral patterns of *ESIM* are reproduced by *CE* in most of the cases, though, not surprisingly, *CE* achieves worse accuracy scores than *ESIM*, as indicated by the difference in scores in the SNLI development set where *ESIM* is better by 0.10 points.

5.9.5 On the Limitations of This Work

Finally, we are aware of limitations in this work that may have influenced the outcomes. First of all, even though our analyses are fine-grained, they are not at the deepest level of granularity. We analyzed both the behavior of the systems and some confounding factors under two experimental conditions, in addition to sub-dividing each one by the type of class label and word pairs; however, we do not account for interactions of these factors. Doing so would lead to a combinatorial number of outcomes (across the levels of each of the factors), probably hazarding the intelligi-

bility of the analyses. We may lose some very fine-grained details, but our current results guarantee internal validity given the high confidence of our statistical tests.

Another limitation is the control of confounding factors. Given that we use original instances from the SNLI dataset as control instances, when we perform our transformations, possible confounding factors arise that are very difficult to control for. For example, intra-sentence word pair interaction; when we move a word from the hypothesis to the premise sentence, and vice-versa, we force this word to interact with the context words surrounding it, and some of these interactions may be new (unseen in the training set.) This factor may, indeed, influence the response of the systems in such a way unknown to us. A possible solution to this problem is to hand-craft instances where we control for this factor; nevertheless, this control would lead to smaller samples than those we use. Furthermore, there are clear advantages of using data from the original dataset, rather than using data crafted under *laboratory* conditions. Even though some factors are left without a proper control, the validity of the results is guaranteed to apply to the original SNLI data, whereas in the case of a *laboratory* sample, the results, though thoroughly validated, are not guaranteed to apply to the data the system was trained with, thus leaving an uncertainty of the extent to which the behavior under study is valid in the *home environment* of the system under study.

Chapter 6

Internal Analysis of *GloVe*: Predicting Hypernymy

6.1 Introduction

Word embeddings are widely used as features in many natural language processing tasks, such as question answering (Kumar et al., 2016), sentiment analysis (Tai et al., 2015), or natural language inference (Parikh et al., 2016) as we saw in the previous chapter. These embeddings are dense vectors that represent concepts in a distributed way. There exists specially dedicated ReLe systems that only learn word embeddings that are then used by other machine learning systems in NLP tasks such as the ones described above. *GloVe* is one of the most popular ReLe systems dedicated to learning word embeddings (Pennington et al., 2014) (for details see Section 6.6.2).

However, because of their opaque nature it remains unclear what linguistic phenomena are captured. A hint of the information learned by *GloVe* word embeddings is given by the objective function used for training the system. This function is designed to capture both a sense of similarity and a sense of juxtaposition among concepts. For example, the concept *canine* is similar to the concept *feline* in the sense that both are animals, but at the same time they differ because they are different types of predators. *GloVe* word embeddings are expected to capture both the similarity and the difference between these two concepts which can be recov-

ered via vector operations over these embeddings. For example, by applying the dot product to the embeddings of *feline* and *canine* we can measure how similar the two concepts are in vector space; similarly, by applying a vector difference to the two embeddings we can measure how different they are.[1] But, as said before, knowing if any other phenomena is encoded in the embeddings is unclear. The ability of uncovering such phenomena not only allow us to better understand the system, but, in addition, to further explore ways of properly exploiting the word embeddings as unsupervised pre-training in NLP tasks (Erhan et al., 2010).

A desirable semantic relation to be encoded in any representation is hypernymy. This relation between concepts provides a hierarchical structure where the concept in the upper level is an abstraction of the concept in the lower level; i.e., the concept below in the hierarchy is a type of the above concept. For example, a *dog* is a type of *canine* which in turn is a type of *animal*. When we speak about abstracting a concept into a more generic concept we refer to a hyponymy relationship, such as when we say that a *dog* is a *canine*. When we ground a concept into a more specific concept then we refer to a hypernymy relationship, such as when we say that *animal* includes the concept of *canine*. In order to analyze if a ReLe system has captured this semantic relation, it is useful to have a taxonomy from which examples can be sampled and thus use them to test the system (or its representations learned.) Nonetheless, creating this type of dataset is not simple; some concepts are *polysemous* (have different senses) and thus difficult to correctly categorize (such as *bank* which can refer to a type of *seat*, a *shore*, or a place where financial operations are carried out); other concepts are also difficult to categorize because their properties do not match all the properties of a category (a *penguin* cannot fly but it is a type of *bird*), and so on. Thus, having appropriate hypernymy data is a crucial aspect for testing whether any system, or representation, has captured this relation.

[1] We note that the values obtained by these two operations are relative rather than absolute. For example, consider the concepts *canine*, *feline*, and *chair*; we would expect that the dot product between the corresponding embeddings of *canine* and *feline* is bigger than that of either *canine* and *chair* or *feline* and *chair*, since *chair* is unrelated to any of the other two concepts. Similarly, we would expect that the vector difference of the corresponding embeddings of *canine* and *feline* is closer to the vector difference of the concepts *dog* and *cat*, rather than to the vector difference of the embeddings of *chair* and *sofa* since these last two concepts are not under a *predator* structure.

Previous work has provided internal analyses of certain ReLe systems at the parameter level –word embedding level– in order to figure out whether the representation learned has captured hypernymy (Weeds et al., 2014; Roller et al., 2014; Vylomova et al., 2016). However, integrating the results obtained across this body of research has been difficult, and previous analysis has shown inconsistencies in the results and has suggested that previous works failed to successfully extract hypernymy information from the embeddings (Levy et al., 2015). Several factors have played a role in the task of extracting hypernym relations, and thus achieving a conclusion has become difficult, and while the nature of these factors is mostly technical, i.e. related to elements in the design of the experiments, the most crucial factor has been the hypernymy data used. Different ReLe systems have been studied under two possible regimes, supervised and unsupervised; different scoring metrics have been used to report results, some of them not entirely appropriate for the task at hand; furthermore, different hypernymy datasets have been used, giving different results, and some of them have been found to contain confounding factors, such as *lexical memorization* which can influence the results to show a misleading picture where it is believed that the system has excelled to learn hypernymy. Hence, the combination across levels of these factors have made it difficult to integrate and interpret the results; though we emphasize on the fact that the datasets play a big role in how the same ReLe system can be pictured as either a failure or a success in capturing hypernymy just by using a different dataset.

In this work, we provide a further internal analysis of a ReLe system with the objective to recover hypernymy from its parameters. But, we propose to do so in a properly controlled experiment where we can eliminate possible sources of confusion. Therefore, we propose to fix to a particular ReLe system, namely *GloVe* (dimensionality $d = 50$), due to its wide use in the NLP community and the availability of its word embeddings. We also propose to use two metrics, namely accuracy and AUC ROC (not used before in the community for this task) appropriate for the task at hand. We choose a supervised approach to recover hypernymy, rather than an unsupervised one, because it has received more attention from the community and,

thus, it is in a more mature state. Having fixed these factors, we propose to study the influence that different datasets have for predicting hypernymy from *GloVe* word embeddings. This study poses some natural questions: How can we study the influence of the different datasets on the results? How can we compare which dataset is more *useful* for extracting hypernym relations from word embeddings? How can we analyze the datasets? I.e. how can we characterize the datasets in order to conclude which one represents a better choice? Can we extract hypernymy from the word embeddings via any of the target datasets? To do the comparison of the datasets, we take motivation from previous work in the computer vision community (Torralba and Efros, 2011) where biases and confounding factors are removed from vision datasets in order to fairly compare them. Hence, we aim to conclude to what extent we are able to extract hypernyms from *GloVe* embeddings, what are characteristics of a good hypernymy dataset, and which dataset from the literature fulfills these characteristics.

6.2 Problem Definition

We aim to know whether *GloVe* word embeddings encode hypernymy information. To test this, we use supervised classifiers that predict if two concepts, represented by word embeddings, are under a hypernymy relationship. The input to the classifiers are only the word embeddings of the two concepts, and the output is the probability of the two concepts being in a hypernymy relationship. However, the experimental design for such a supervised task can be easily affected by some factors, where the choice of dataset may be the most important one. Therefore, we define our problem as follows. Given both the embeddings learned by *GloVe* and a set of hypernymy datasets, we aim to provide controlled experiments where we can study how the datasets influence the decision of the classifiers, and how the correct choice of a dataset can allow us to conclude whether *GloVe* embeddings have captured hypernymy information or not. To do this, we propose the following setup. First, we propose to use two classification measures widely used in other machine learning communities for evaluating classifiers. Second, we propose to study the influence

of six different hypernymy datasets on the classifiers' ability to recover hypernymy from word embeddings; this study is carried out by comparing in a *fair way* how well the classifiers can extract hypernymy given each of the datasets. Third, we propose to analyze why a dataset may or may not be useful for extracting hypernymy, and so, explain the results from the comparison of the datasets.

6.3 Research Questions and Hypotheses

Our central research question is about understanding what information has been learned by the *GloVe* system. More concretely, we ask whether *GloVe* has captured hypernymy information in its parameters –word embeddings. Our interest in this question relies in the fact that the objective function for training *GloVe* does not have an extrinsic component dedicated for learning hypernymy; thus, a controlled study is the only way to answer our central question. With this question in mind, and taking into consideration the results from previous work where no consensus has be achieved, we pose the following questions.

Research Questions

1. How can we fairly compare existing hypernymy datasets?

2. What confounding factors are to be controlled in order to compare the datasets? I.e. What are possible confounding factors in the datasets that may influence the decision of the classifiers?

3. How can we control for such confounding factors?

4. Are accuracy and AUC ROC (Area Under the Curve of the Receiver Operating Characteristic) appropriate measures for evaluating classifiers in this task?

5. What characteristics make a dataset consistent and useful for recovering hypernymy from word embeddings? I.e. how can we characterize a hypernymy dataset?

    - How can we measure these characteristics in a dataset?

- How can we corroborate the value of these characteristics?

Hypotheses

1. We hypothesize that both accuracy and AUC ROC are suitable metrics for evaluating binary classifiers, as previous work in the machine learning community has shown.

2. We hypothesize that one possible confounding factor, previously exposed by Torralba and Efros (2011) for comparing visual datasets, has to be controlled in order to compare the hypernymy datasets, namely the dataset size.

3. We also hypothesize that an appropriate way of comparing the datasets, as proposed by Torralba and Efros (2011), is via *cross-test* evaluations where a classifier trained on a dataset D_i is tested not only on the test set of D_i but also on the test sets of other datasets.

4. Based on previous work in psychology (Murphy, 2004), and based on our own observations of hypernymy data, we hypothesize that any consistent and useful hypernymy dataset should encode two important properties for the target phenomena, which we call *generality* and *similarity*.

6.4 Contributions

- Our main contribution is focused on understanding a specific aspect of ReLe systems, the ability to learning a linguistic phenomena as a side effect of the whole learning procedure. In this sense, we provide evidence towards *GloVe* word embeddings encoding hypernymy information.

- To achieve our main contribution, we provide a controlled comparison of hypernymy datasets. We study six commonly used hypernymy datasets in order to figure out if any of them is a useful dataset to recover hypernymy.

- We also analyze the characteristics that we propose that make a hypernymy dataset useful for recovering hypernymy.

- Based on our analysis, we provide a way to improve hypernymy datasets, i.e. we develop a sampling procedure based on the characteristics of hypernymy that allows a classifier to improve its accuracy score.

- Finally, we analyze to what extent accuracy and AUC ROC are suitable for assessing classifiers in the task of hypernym extraction.

6.5 Scope and Limitations

We study one specific ReLe system, namely *GloVe*, which means that our results may not generalize to other ReLe systems. Furthremore, we study whether *GloVe* has captured a specific phenomena, namely hypernymy. Thus, the conclusions that we achieve may not apply to any other semantic relation, such as antonymy or meronymy. In addition, we are aware that we did not study all the existing hypernymy datasets in the literature; we targeted those most commonly used, but we leave for future work the analysis of the rest of the datasets. Furthermore, our analysis of hypernymy datasets may provide a schema (or at least an idea) of how to analyze other types of datasets, but to what extent this is the case is unknown to us and such endeavour is out of the scope of this work.

6.6 System Under Study

In this section we describe *GloVe*, our target system, and the data used for training it.

6.6.1 Wikipedia and Gigaword Data

The version of *GloVe* that we analyze was trained using a combination of two datasets, namely Wikipedia 2014 and Gigaword 5 (Parker and et al., 2011). The first dataset is the collection of Wikipedia articles from 2014 (all available documents in English.) The second dataset is a collection of English news text from seven sources, namely the Agence France-Presse, the Associated Press Worldstream, the Central News Agency of Taiwan, the Los Angeles Time/Washington Post Newswire Service, the Washington Post/Bloomberg Newswire Service, the New York Times

Newswire Service, and the Xinhua News Agency; this collection was done by the Linguistic Data Consortium from the University of Pennsylvania.

6.6.2 *GloVe*: Global Vectors

GloVe (Pennington et al., 2014) is a ReLe system that learns word embeddings based on the auxiliary task of predicting occurrence counts of pairs of words; its formulation is similar to that of a matrix factorization model (Section 2.1.1.1.) The objective function it minimizes is the following:

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{|V|} f(X_{ij})(\mathbf{w}_i^T \tilde{\mathbf{w}}_j - logX_{ij}) \tag{6.1}$$

This equation is an approximation of the ratio of co-occurrence probabilities:

$$\frac{p(word_k|word_i)}{p(word_k|word_j)} \tag{6.2}$$

Where $word_k$ is for all words in the vocabulary; the function $f()$ in Equation 6.1 is a weighting function; it gives higher weight to more frequent word pairs than to rare ones. $\mathbf{X}_{n\times n}$ is a matrix of co-occurrences (rows and columns correspond to words in a vocabulary $\mathbf{V}$ of size n), and, therefore, X_{ij} is the count of word w_i co-occurring with word w_j in a corpus. The parameters are the word embeddings $\mathbf{w}_i$ and $\mathbf{w}_j$. We notice that in Equation 6.1 the tilde on the embedding of word w_j indicates that this embedding comes from a set of embeddings associated to the columns of matrix $\mathbf{X}$, whereas the embedding of word w_i comes from a set of vectors associated to the rows. This design –having one set of embeddings for columns and one set for rows– corresponds to a matrix factorization model.

After the word embeddings are learned, via a gradient descent algorithm, they are ready to be used as features in a downstream task. An excerpt of the word embedding learned for the concept *cat* is [0.45281 -0.50108 ... 0.71278]; as we see, an embedding is a continuous vector of fixed dimension. Since two sets of embeddings are learned for the same words, we either take only one set or linearly combine both sets.

GloVe embeddings evaluated on the tasks of similarity and analogy provided

state-of-the-art results on the datasets of *wordsim353* (Finkelstein et al., 2002) and (Mikolov et al., 2013) at the time of publication of the corresponding article. These results seem to indicate that the vector space of GloVe captures some semantic information.

6.7 Methods and Materials

6.7.1 Data

Here we describe several aspects related to our analysis of hypernymy datasets. First, we describe what is a hypernymy dataset (Section 6.7.1.1); then, we explain a confounding factor found in some datasets, namely *lexical memorization* (Section 6.7.1.2); after that, we describe the six hypernymy datasets that we analyze (Section 6.7.1.3); finally, we describe how we control for some factors on the datasets in order to produce *controlled* datasets to fairly compare them for the task of hypernymy extraction from *GloVe* embeddings (Section 6.7.1.4).

6.7.1.1 A Hypernymy Dataset

An instance in a hypernymy dataset is of the form $(c_i, c_j, label)$, where c_i and c_j each refer to a concept, and *label* refers to whether the two concepts are in a hypernym relation. We say that a concept c_i takes the position of a hyponym and c_j takes the position of a hypernym. For example, *(dog, animal, True)* is a positive instance where *dog* is positioned in the slot corresponding to a hyponym and *animal* is positioned in the slot corresponding to a hypernym; thus, this positive instance can be read as *a dog is a type of animal*, which is a true statement. Examples of negative instances are the following:

(6.3) *(car, animal, False)*

(6.4) *(animal, car, False)*

(6.5) *(dog, cat, False)*

(6.6) *(dog, invertebrate, False)*

(6.7) *(animal, dog, False)*

These are negative instances since they do not fulfill a hypernymy relation. For example, in Example 6.3 we can clearly recognize that a *car* is not a type of *animal*; we also recognize that a *dog* is not a type of *invertebrate* (Example 6.6); and the same with the rest of the examples. We now categorize in a fine-grained detail these types of negative instances according to our hypothesis that there are two important properties for characterizing hypernymy, namely *generality* and *similarity*.

These negative instances differ along three dimensions, two of them mentioned above (*generality* and *similarity*), and the position that a concept takes within the instance (hyponymy or hypernym). In the negative instance of Example 6.3, *car* is less abstract (general) than *animal*, and both concepts are dissimilar from each other. In Example 6.4, similarly to the first example, the relations between both concepts are the same, but their order within the instance is different; in these two examples the order of the concepts does not matter, since they do not fulfill the similarity constraint, i.e. they are unrelated concepts. In Example 6.5, both concepts are similar but none of them is more abstract than the other. In Example 6.6, *dog* is less abstract than *invertebrate*, and both are similar to some degree, but, indeed, a dog is a *vertebrate*. Finally, in Example 6.7, we see a true hyponymy relationship, and not a hypernymy relationship, due to the order of the two concepts.

We note that the datasets under study have similarities and dissimilarities. First, all the datasets contain the same type of positive instances since there is only one type, as shown above, but they may not contain all the types of negative instances we described. Furthermore, the sources where the instances come from, the sampling process, and the motivation for creating the datasets are different, and therefore, the resulting datasets differ mainly in sample size, in the ratio of positive vs. negative instances, and the quality of the instances (some datasets contain more noisy instances than others which affect the performance of the classifiers.)

We also note that since some datasets were constructed with a different objective from that of studying hypernymy, these datasets had to be tailored for the task of predicting hypernym relations. Therefore, we use the tailored version of those datasets as provided by Levy et al. (2015). In the following sections we describe

both the original datasets and how they were tailored for hypernymy prediction. Table 6.1 shows a summary of the datasets. These datasets, in turn, will be modified according to our experimental design in order to control for two confounding factors, namely *sample size* and *imbalance* (Section 6.7.1.4.)

6.7.1.2 Lexical Memorization as a Confounding Factor

Previous work exposed a confounding factor lying in some hypernymy datasets, namely *lexical memorization* (Weeds et al., 2014; Levy et al., 2015; Vylomova et al., 2016). If a concept c_j is mainly seen in the position of a hypernym concept across positive instances, as in $(c_i, c_j, True)$, then the classifier may memorize that c_j is a hypernym of the concept in the hyponym position, and thus it may predict a true hypernym relationship between the two concepts regardless of the information contained in the word embeddings representing these two concepts. To control for this factor, previous work has suggested having disjoint vocabularies between training and test sets. In this way, the concept c_j from the example above will never appear on the test data and thus cannot influence the score of the classifiers.

6.7.1.3 Hypernymy Datasets From the Literature

Baroni Dataset Baroni et al. (2012) extracted 9734 positive examples of hypernymy from WordNet; all of these examples were manually checked to discard noisy ones. After this selection process, they kept 1385 examples. They created two types of negative instances; on the one hand, they selected 33% of the positive instances and swapped the concepts in order to obtain hyponym pairs, i.e. the concept in the hyponym position was swapped with the concept in the hypernym position; the type of negative instance obtained with this swap is similar to Example 6.7. On the other hand, they randomly paired concepts from different positive instances, thus obtaining instances similar in form to the Examples 6.3 and 6.6. At the end, Baroni et al. (2012) obtained 1385 negative instances, all of them manually checked in order to discard any positive instance. Since the number of positive and negative instances is the same, this dataset is balanced. In order to control for the lexical memorization factor, Levy et al. (2015) deleted those instances that had a concept seen in training data and a concept seen in test data. The sample size of the resulting training set

is 791 instances where roughly 49% are positive instances, and the test set contains 536 instances almost perfectly balanced between positive and negative.

Bless Dataset Baroni and Lenci (2011) created a dataset with the objective of semantic evaluation for distributional models. This dataset includes instances of several semantic relations such as meronymy (for example, *(wheel, car)*, since one is a part of the other), hypernymy, co-hyponym (for example, *(cup, glass)*, since both are sister concepts in a taxonomy), random (for example, *(car, bird)*, since both concepts have no relation with each other), among others. Most of the 200 concepts filling the hyponym position in an instance come from the McRae dataset (McRae et al., 2005). These concepts are distributed among 17 classes of concepts that include both living and non-living entities (vehicle, animal, etc.). Also, these concepts are neither ambiguous nor highly polysemous. The concepts in the hypernym position within the instances come from several resources such as the McRae dataset, WordNet (Fellbaum, 1998), ConceptNet (Liu and Singh, 2004), and Wikipedia, and were validated by the authors. The concepts, for instances of the random relation, were randomly drawn from the positive instances of the other relations; then, crowdsource annotators confirmed whether there was a truly random relation between the two concepts. In order to obtain a hypernymy dataset, Levy et al. (2015) used the instances of the hypernymy relation as positive instances and the positive instances from meronym, co-hyponym, and random relations as negative instances. The sample size of the train set is 3225 instances with approximately 90% negative ones. The test set contains 3650 instances where 92% are negative ones. The resulting training and test sets have disjoint vocabularies.

Kotlerman Dataset This dataset is a version of the lexical entailment dataset of Zhitomirsky-Geffet and Dagan (2009). This dataset was created with the purpose of evaluating directional-distributional similarity measures. Such a measure determines whether two concepts (c_i, c_j) hold in an entailment relation, in the sense that a concept c_j is implied in the meaning of a concept c_i; for example, *(chess, game, True)* and *(divorce, marriage, True)* are positive instances of entailment (Kotlerman et al., 2010). In addition, synonyms are considered as two-way positive instances,

such as *(car, automobile, True)* and *(automobile, car, True)*. Each instance in the dataset was obtained using the *LIN* measure (Lin, 1998) which computes the similarity of two concepts based on their distributional vectors. The similarity is proportional to the number of features both concepts have in common. The instances obtained by this measure were then manually annotated as either positive or negative leading to a sample size of 3772 instances. Levy et al. (2015) obtained disjoint training and test sets using those instances from which 739 instances were kept (around 69% are negative instances.) The test set contains 621 instances with 26% positive instances.

Levy Dataset Levy et al. (2014) gathered a set of 68 million propositions in the form *(subject, verb, object)* from Google's syntactic n-grams, for example *(aspirin, cure, headache)*. They kept propositions from the health-care domain and manually discarded noisy ones. The remaining triples were manually annotated by native English speakers for the task of entailment. From the set of entailing propositions, Levy et al. (2015) extracted entailing nouns that shared two arguments while keeping a disjoint vocabulary between training and test sets. The meaning of entailment between two propositions applied by (Levy et al., 2014) is based on the ability of a human to decide whether one proposition is true given the other one. The resulting training set consists of 2932 instances where only 8% are positive. The sample size of the test set is 2985 instances with 93% being negatives.

Turney Dataset Turney and Mohammad (2015) created this dataset in order to evaluate lexical entailment. In order to know whether a concept c_i entails another concept c_j, a semantic relation bounding the two concepts should be taken into consideration as background knowledge; thus, entailment is derived from the meaning of this semantic relation, provided that there is little ambiguity in the meaning of the concepts, i.e. they are not highly polysemous. Then, Turney and Mohammad (2015) transformed the SemEval-2012 dataset to expand from 79 to 158 semantic relations by taking the reversed relations as new types of relation. Both symmetric and asymmetric relations were considered, including synonymy. Then, Turney and Mohammad (2015) manually annotated each relation as either a positive or negative

entailment. Levy et al. (2015) obtained a disjoint training set comprising 539 pairs from which roughly 52% are positive. The test set has 507 pairs where 47% are positive.

Weeds Dataset Weeds et al. (2014) created a dataset from WordNet for hypernym detection. The design of this dataset controls for other possible confounding factors besides lexical memorization, namely 1) imbalance, 2) preventing the classifiers from predicting hypernymy based only on the similarity of the concepts, 3) preventing the classifiers from learning a taxonomical relationship among three or more concepts just by the arrangement of the instances, for example by virtue of having the instances *(dog, canine, True)*, *(canine, vertebrate, True)*, and *(vertebrate, animal, True)* in the training set a classifier may discover that *(dog, animal, True)* is a correct instance without looking at the word embeddings of the concepts and only from the disposition of the instances.

Thus, Weeds et al. (2014) built the dataset in the following way. They selected those words that were 1) frequent (according to some threshold) in the WordNet SemCor package and Wikipedia and 2) monosemous, according to the frequency of its predominant sense in the WordNet SemCor package. Thus, polysemous words were filtered out. Weeds et al. (2014) aimed for a balanced dataset where half of the negative instances were co-hyponyms[2] and the other half were positive instances reversed. A constraint for accepting any instance, positive or negative, from WordNet to populate the dataset was that each concept in such an instance has not been previously seen in the same position (hyponym or hypernym) in any other instance already selected. The resulting training set contains 2033 instances balanced between positives and negatives. The test set is of size 123 instances, balanced as well.

[2]A co-hyponym pair (c_i, c_j) is one where both concepts c_i and c_j share a common hypernym concept, i.e. both have the same *parent* in a taxonomy tree.

Dataset Name	Training Set Size	Ratio positive/negative instances
Baroni	791	0.97
Bless	3225	0.12
Kotlerman	739	0.45
Levy	2932	0.08
Turney	539	1.06
Weeds	2033	0.98

Table 6.1: Summary of datasets. *Training Set Size*: Number of instances (positive and negative).

6.7.1.4 Adjusted Hypernymy Datasets for Extracting Hypernymy from *GloVe* Embeddings

We take the datasets from Section 6.7.1.3 and adjust them to control for two factors: Sample size, similar to (Torralba and Efros, 2011), and imbalance. In other words, we aim to have comparable datasets to fairly compare them for the task of hypernymy extraction. Hence, we want the training sets of each dataset to contain the same number of instances; and each training set should contain the same number of positive and negative instances. These adjustments allow us to fairly compare the datasets in terms of their usefulness, or quality, to predict hypernym relations. A training set with a bigger sample size may be deemed more useful than one with fewer training instances since the confidence, in terms of error, for parameter estimation of the classifier may be better; however, the opposite scenario may occur if many of the instances from the bigger dataset are *noisy* or incorrect, such as *(dog, animal, False)*. Thus, in order to evaluate how useful the datasets are in terms of the quality of the instances rather than their sample size, we fix them all to have the same number of instances. On the other hand, an imbalanced dataset biases a classifier towards predicting the label of the majority class, thus giving an over-optimistic result on the classification task. One way to overcome this problem is to choose a metric score, for evaluating the classifier, that focuses on the minority class, such as the F_1 measure; however, we opt to use two other metrics widely used for evaluating binary classifiers in the machine learning community, namely AUC ROC and

accuracy, both of them requiring a balanced dataset.

We do the adjustments proposed above to the training sets (and a similar adjustment to the test sets) of each dataset. In order to obtain both an aggregated measure (average score) and a confidence value (standard error) to evaluate the classifiers trained with the adjusted hypernymy datasets, we obtain 20 bootstrapped, adjusted training sets from each dataset. To do so, we uniformly at random sample instances with replacement[3] from each of the original training sets in Section 6.7.1.3 to generate its corresponding 20 training sets. All of these resulting training sets are both normalized to the same sample size and balanced. (Then, for example, we will use 20 normalized and balanced training sets generated from the Baroni dataset to train 20 Baroni classifiers to average their performance.) We choose a sample size of 400 instances, and so, we end up with 200 positive and 200 negative instances in each training set.

On the other hand, we adjust the test sets by balancing them but we do not normalize them. We balance the test sets to match the number of instances in the majority class to the number of instances in the minority class by discarding some instances from the majority class; since each of the original test sets has a different sample size, the resulting test sets are not normalized to the same number of instances. The reason for not normalizing the test sets is because it is not necessary; due to the evaluation methodology, which we explain in Section 6.7.2, sample size does not play a role as a confounding factor in the test sets.

In both cases, for adjusting training and test sets, we apply an *under-sampling* scheme to the datasets in order to obtain smaller training and test sets. This sampling scheme, compared to an *over-sampling*[4] scheme, has the advantage of avoiding overfitting because instead of copying instances it discards instances (He and Garcia, 2009). Nonetheless, we also try an *over-sampling* scheme for balancing the test sets, though our results are very similar to the *under-sampling* scheme and so

[3]That is, all the instances have the same probability of being selected and can be sampled more than once.

[4]In this scheme, the aim is to match the number of instances from the majority class; to do so, one needs to over-sample instances from the minority class. In this way, multiple copies of the minority class instances are generated.

we will not report these results.

To summarize, we obtain 20 under-sampled training sets from each of the datasets in Section 6.7.1.3. All the training sets contain 400 instances balanced between positive and negative labels. The test sets are also under-sampled in order to be balanced, but they are not normalized as the training sets. The number of instances in each test set depends on the number of positive instances; we discard negative instances (majority class) in order to match the number of the positives. This adjustment process helps us to both fairly compare how useful the datasets are for learning to classify hypernyms and testing the classifiers using appropriate metrics, namely AUC ROC and accuracy.

6.7.2 Cross-test Evaluation

To elucidate whether *GloVe* word embeddings capture hypernymy information, we compare the six datasets from Section 6.7.1.3 by using the adjusted datasets obtained in Section 6.7.1.4 which do not contain confounding factors that may influence our outcome. We apply the cross-test evaluation from Torralba and Efros (2011) to do the comparison. In this evaluation, we train classifiers using each of the adjusted training sets and we test them on each of the adjusted test sets derived from the six original datasets. In this way, given that for each of the six original datasets we obtain 20 adjusted training sets, we end up with 20 classifiers, which, in turn, are evaluated on each of the adjusted test sets. Therefore, each original dataset is evaluated through the ability of 20 classifiers to generalize to its own test set and to the other five test sets.

In order to clarify the method proposed above, we take the evaluation of the Baroni dataset as an example. We generate 20 adjusted training sets and one adjusted test set from this dataset (likewise for the rest of the datasets.) We obtain one classifier for each of the adjusted Baroni training sets. Each classifier is tested on the Baroni adjusted test set, and on the adjusted test sets of the other five datasets. Finally, we average the scores obtained from the 20 the classifiers; thus to evaluate the Baroni dataset, we report the average score of the 20 Baroni classifiers on the adjusted Baroni test set, the average score of the 20 Baroni classifiers on the

adjusted Bless test set, and so on.

The results of a cross-test evaluation can tell us how useful is a dataset for a classifier to generalize to different distributions of data. Each dataset is obtained from a particular source of information, for example WordNet, which may or may not be representative of the phenomenon under study, namely hypernymy. Thus, a dataset that allows a classifier to generalize beyond its own distribution of data can be regarded as a good dataset. On the contrary, a dataset that leads a classifier to perform poorly on other test distributions can be discarded; such poor results may mean that the dataset is not representative of the phenomenon or that a considerably large portion of the instances are ill-built.

Based on this type of evaluation of a dataset, natural questions arise regarding confounding factors and explanations of the data itself: Does the choice of classifier play a role in the generalization ability? What if we have chosen a classifier suitable for a specific dataset but not for another one? What are the correct metrics to evaluate the classifiers? Can we expect a certain type of structure in the dataset that allows a classifier to generalize the best?[5]

Answers to the questions above are as follows. In order to control for any possible effect of the learning mechanism underlying the datasets, namely the classifier and the vector operation over the embeddings, we try, for each adjusted training set, each possible combination of classifier (logistic regression, support vector machine.[6] For a description of these classifiers see Sections 2.2.1 and 2.2.2) and vector operation (*diff*, *concat*). We use the best result on validation data to select the classifier and the vector operation to be evaluated on test data. We choose two score metrics to evaluate each classifier on test data: AUC ROC and accuracy, explained in the following sections.

Finally, as for our last concern, we hypothesize that a good dataset follows a structure according to a background theory of the phenomenon under analysis; in our case, we hypothesize that the background knowledge required to create a

[5] Another possible factor is the presence of out of vocabulary words; for those rare concepts for which there are no word embeddings associated we use the *unk* embedding, an embedding specially used for out of vocabulary concepts.

[6] We also try each possible kernel among linear, polynomial, and RBF.

hypernymy dataset are two conditions proper of hypernymy: We characterize this linguistic phenomenon along two dimensions, generality and similarity. In Section 6.7.3 we formalize our analysis of this background theory in order to experimentally prove our hypothesis.

6.7.2.1 Ranking Instances: AUC ROC

Previous work in hypernymy prediction has mainly used F_1 (see Equation 6.8) as the metric for evaluating hypernymy classifiers. However, we find two characteristics of this metric not suitable for our purposes. First, its usefulness for the task of comparing classifiers has been questioned (He and Garcia, 2009). Second, F_1 works on predictions thresholded at $\alpha = 0.5$, an un-optimized threshold, thus loosing information of the classifiers' behavior on other thresholds. Instead, previous work has suggested using AUC ROC (Huang and Ling, 2005) for a better evaluation of classifiers.

$$F_1 = 2\left(\frac{Precision * Recall}{Precision + Recall}\right) \tag{6.8}$$

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{6.9}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{6.10}$$

AUC ROC provides a single score for a classifier that summarizes the trade-off between true positive and false positive rates (TPR and FPR) across several thresholds on the predictions (Equations 6.11 and 6.12.) AUC ROC stands for Area Under the Curve of Receiver Operating Characteristics, where ROC is a graph (FPR vs. TPR) across several thresholds of the predictions, i.e. each point in the graph represents a trade-off between false positives and true positives for a given threshold. The higher the AUC the better the ability of the classifier. A qualitative interpretation of the AUC ROC score is the evaluation of a classifier's ability to rank positive instances with respect to negative ones (Huang and Ling, 2005). In addition, it has been shown that AUC ROC is equivalent to the Wilcoxon sum rank test (Huang

and Ling, 2005), a nonparametric statistical test suitable for comparing classifier's ranking abilities (Perlich et al., 2003).

$$TPR = \frac{TruePositives}{TotalPositives} \tag{6.11}$$

$$FPR = \frac{FalsePositives}{TotalNegatives} \tag{6.12}$$

However, previous work has pinpointed some drawbacks of using AUC ROC in highly imbalanced datasets. When the number of negative instances is considerably bigger than that of positives, the true negatives predicted by the classifier will help *push* the true positives in the ranking and thus will lead to a good AUC ROC score (Zou et al., 2016). Furthermore, even if the presence of false positives is comparable to that of true positives, this will not seriously impact the AUC ROC score, due to the high imbalance in the data (Zou et al., 2016; Lobo et al., 2008). To deal with this problem, we balance the test sets; as described in Section 6.7.1.4, we use an *under-sampling* scheme (which gets similar results compared to an *over-sampling* scheme.)

Thus, our experiments in Section 6.8 provide a cross-test evaluation reporting AUC ROC scores, from which we can elucidate to what extent each dataset helps classifiers to rank positive hypernymy relations with respect to negative relations using *GloVe* embeddings.

6.7.2.2 Predicting Hypernym Relations: Accuracy

As a complementary measure to AUC ROC, we provide accuracy scores for the cross-test evaluation of the classifiers. This metric can tell us to what extent the datasets are useful for classifiers to predict hypernym relations. We use the adjusted test sets from Section 6.7.1.4 to compute these scores (the same test sets that we use for computing the AUC ROC scores.) Since these test sets are balanced, we do not have the common problems found when computing accuracy on imbalanced datasets, as previous work has shown (Huang and Ling, 2005). The main advantage of using accuracy scores instead of F_1 scores on a balanced dataset is the easiness

of interpretation of results; in a binary classification problem using accuracy, it is clear that a classifier that predicts all labels to be in the same class will achieve a 50% score, whereas using F_1 the score is around 0.66, a less intuitive score. Thus, we stick to accuracy for reporting hypernymy prediction. We optimize a threshold for each classifier on validation data.

6.7.3 Dataset Analysis

In order to understand why a dataset D_i is more useful than a dataset D_j for extracting hypernymy relations from *GloVe* word embeddings, we analyze the datasets under a background theory, i.e. the conditions we believe to be pertinent for characterizing hypernymy: *Generality* and *similarity*. We base our background theory on previous theories of concepts from psychology (Murphy, 2004).[7] In a taxonomy, *generality* refers to the level of abstractness of a concept compared to other concepts; i.e., it denotes that a concept c_j includes another concept c_i if the latter includes all the features of the former. *Similarity* refers to the degree of overlap of features a concept c_j has with another concept c_i. For example, if we compare *dog* with *cat* under the concept of *animal*, we find out that both are sibling terms; none of them is a more general (abstract) concept than the other. A *cat* has properties that a *dog* does not have, like whiskers, and vice-versa, *dog* has features not included in the definition of *cat*; so, none of them encompasses the other, but both concepts share many features proper of any *animal*, thus they are similar to each other.

We analyze the six datasets from Section 6.7.1.3 along the dimensions of *generality* and *similarity*: Do the instances in the dataset fit our background theory? I.e., do the datasets encode these two characteristics? Are these two characteristics relevant for a hypernymy dataset? We respond to these questions via two experiments. In the first one, we explore the datasets in order to analyze patterns of generality and similarity. In the second experiment, we provide a mechanism in order to cre-

[7]Previous theories have claimed that people categorize objects based on how similar they are to common objects pertaining to a category; for example, when a person sees a dog in the street she may classify it as such by comparing it to her dog at home. Claims have also been made about people learning hierarchies of concepts based on common features across objects; for example, when a person describes an animal, she may mention general attributes present in all animals, such as breathes and eats, but when she describes a more specific animal, such as a dog, then she may mention attributes more specific, such as barks.

ate hypernymy datasets that leverage generality and similarity patterns to figure out to what extent these two patterns are useful.

6.7.3.1 Exploration of Datasets: Recovering Generality and Similarity Patterns

We operationalize our background theory based on WordNet (Fellbaum, 1998) given that it is a taxonomy encompassing a wide range of concepts and it has been widely used in the community. We define *generality* as a function, $g()$, over two concepts, c_i and c_j, where the resulting value, $g(c_i, c_j)$, is the difference in abstraction between the two concepts; i.e., *generality* is the difference in the number of edges in WordNet with respect to the root of the taxonomy, as shown in Equation 6.13, where the function $distance(c_i, c_j)$ is the number of edges separating concept c_i from concept c_j. We note that the closer a concept is to the root of the WordNet taxonomy, the more abstract it is.

On the other hand, we define *similarity* using the Wu-Palmer similarity function (Wu and Palmer, 1994), as shown in Equation 6.14, where *LCS* stands for *least common super-concept*, which is the closest common ancestor concept closest to both concepts c_i and c_j, and the function $num\_nodes(c_k, c_l)$ returns the number of intermediate concepts in the taxonomy that connects two concepts c_k and c_l. We note that the Wu-Palmer function values are in the interval $[0.0, 1.0]$ where the maximum value of similarity is given only when c_i and c_j are synonyms.[8] However, in our experiments we re-scale these values to the interval $[-1.0, 1.0]$ for visualization purposes. Thus, a similarity value of $s(c_i, c_j) = -1.0$ can be interpreted as the two concepts being totally unrelated, whereas the value $s(c_i, c_j) = 1.0$ imply the two concepts being synonyms. Similarity values in the range $[-1.0, 0.0]$ can then be interpreted as levels of unrelatedness, while those in the range $[0.1, 1.0]$ can be regarded as levels of relatedness.

$$g(c_i, c_j) = |distance(c_i, root) - distance(c_j, root)| \tag{6.13}$$

[8] More specifically, a similarity value of $s(c_i, c_j) = 1.0$ means that both concepts make reference to the same *synset* in WordNet, i.e. they have the same meaning.

$$s(c_i, c_j) = \frac{2 \times num\_nodes(LCS, root)}{num\_nodes(c_i, LCS) + num\_nodes(c_j, LCS) + 2 \times num\_nodes(LCS, root)} \quad (6.14)$$

An example of *generality* and *similarity* functions applied to the concepts *animal* and *cat* is the following. The difference in generality in WordNet between *animal* and *cat* is $g(animal, cat) = |6 - 13| = 7$; this means that *animal* is more general than *cat* by several levels of abstraction. Their similarity, in terms of the Wu-Palmer function, is $s(animal, cat) = 0.32$, which can be interpreted as both concepts being somewhat similar. If we apply *generality* and *similarity* now to the sister concepts (co-hyponyms) *terrier* and *hound* we obtain the values $g(terrier, hound) = 0.0$ and $s(terrier, hound) = 0.62$. We obtain a generality value that indicates that both terms are in the same level of abstraction since they have a common concept parent, *hunting dog*; i.e., there is only one edge from each concept to their parent concept. The similarity value indicates that both concepts are, indeed, very similar to each other since both are types of dog. From these two examples we can see that certain patterns follow after some types of concept pairs; for example, generality and similarity values are specific for co-hyponyms, as we just saw.

We now propose a structure expected in a hypernymy dataset based on the characteristics of *generality* and *similarity*. Assuming a hypernymy dataset is sampled from a taxonomy, where each instance is drawn uniformly at random, we would expect to draw more concepts from the bottom than from the top of the taxonomy. The rationale behind this assumption is because a concept at the level α in the taxonomy has, in average, β edges that lead to the $\alpha + 1$ level, a lower level in the taxonomy; thus, if in the α level there are k concepts, then in the $\alpha + 1$ level we expect that there are $k \times \beta$ concepts. Therefore, if all concepts at both levels, α and $\alpha + 1$, have the same probability of being drawn, it is more likely to draw a concept from the lower level than from the upper level. Following this logic, we expect that most of the instances, positive and negative, will have a small generality value $g(c_i, c_j)$; thus we expect that across the possible generality levels, the number of positive and

negative instances will decrease as we move towards the highest level. As we saw in the first example above, the difference in generality between *animal* and *cat* is high because the former concept is much closer to the root of the taxonomy than the latter concept. And since we expect to sample highly abstract concepts less times than more grounded concepts, like *cat*, such high generality values should be seen less often than small values. We note that the lowest level in generality, $g = 0$ where both concepts are in the same level of the taxonomy, is devote only to co-hyponyms, a type of negative instance, as we saw in the second example above.

We now describe similarity patterns to be found in the data. First of all, we note that similarity values of most of the positive instances should be in the range $[0.1, 1.0]$[9] denoting a level of relatedness between the concepts in the instance. On the other hand, similarity values of negative instances can be spread along the whole range $[-1.0, 1.0]$ due to the variety of possible negatives. For example, we saw before that the instance *(animal, cat, False)* has a similarity value of $s(animal, cat) = 0.32$; this is a similarity value that denotes relatedness because this instance is the reversed of the positive instance *(cat, animal, True)*, and the Wu-Palmer function is symmetric. A similar scenario can be put for a negative instance containing co-hyponyms, as we saw before with the example *(terrier, hound, False)*. Similarity values denoting unrelatedness can be found in negative instances where the concepts are unrelated; for example, the instance *(cat, knife, False)* has a similarity value of $s(cat, knife) = -0.4$. Thus, we can expect negative instances to populate all similarity levels, while positive instances should only populate what we call relatedness levels.

A second similarity pattern describes the mass of the positive instances being concentrated around high levels of similarity. This is because, following the rationale used above to describe generality patterns expected in the data, we expect the data to contain more positive instances with concepts from the bottom of the taxonomy having a close common ancestor, and thus being highly similar to each other, than concepts from the top of the taxonomy. Thus, we would expect that pos-

[9] Except for a few instances where the hyponym is at the bottom level and the hypernym is the root of the taxonomy.

itive instances decrease in number as the similarity levels also decrease. In the case of negative instances, we would expect a more uniform distribution of instances than that of the positive instances because of the variety of the types of negatives. The exact number of negatives in the low levels of similarity ($s \leq 0.0$) and on the high levels depend on the sampling procedure used to create a dataset; we expect that the majority of negative instances contain concepts from the bottom rather than from the top of the taxonomy, and if so then such concepts are either unrelated or co-hyponyms with similarity values falling mostly in the low and high levels, respectively.

6.7.3.2 Dataset Creation: Leveraging Generality and Similarity Patterns

As a proof of concept of our hypothesis that the characterization of hypernymy along *generality* and *similarity* dimensions is important for creating a hypernymy dataset, we create a new dataset following a structure, in terms of generality and similarity patterns, similar to the structure described in Section 6.7.3.1. More concretely, we copy the structure from the dataset that produces the best experimental outcomes (AUC ROC and accuracy scores) from the cross-test evaluations described in Section 6.7.2.

We create rules in order to copy the structure from the most useful hypernymy dataset; with these rules we try to distribute positive and negative instances as similar as possible as they are distributed in that dataset. The rules that we follow in order to accept an instance $h = (c_i, c_j, label)$ to populate our new dataset have the structure: IF generality level is $g = i$ AND the ratio of positive vs. negative instances in this generality levels is $r_g = m$ AND number of instances in this level is $<$ threshold t_g THEN accept instance h to populate dataset. And similar rules across similarity levels. We note that we design these rules by hand.

We create 20 balanced training sets, each of size 400 instances (200 positive and 200 negative instances) by bootstrapping. We sample both positive and negative instances from the union of the training sets of the six datasets described in Section 6.7.1.3. We sample each instance uniformly at random and if that instance fulfills

our rule described above then we accept it to populate our new dataset. We compare the usefulness of this new dataset against two baseline datasets. The first one is created in the same way as our proof-of-concept dataset except that we do not use any rules for accepting an instance, i.e. any instance that is sampled is accepted to populate the baseline dataset. The second dataset is created in the same way as our proof-of-concept dataset, except for the rules to follow in order to accept an instance; we design rules to copy the structure of the least useful dataset (according to results from cross-test evaluations), and then we follow those rules to generate the new dataset. These two baseline datasets are also bootstrapped to be compared against our proof-of-concept dataset.

Thus, we end up with three datasets, namely our proof-of-concept dataset, that we believe to be useful for extracting hypernymy from word embeddings, and two baseline datasets against which we compare the former. To do so, we perform cross-test evaluations where we train classifiers using these three new datasets and test them on the adjusted test sets from Section 6.7.1.4. We report average accuracy scores across the 20 training sets of each dataset.

6.8 Experiments and Results

6.8.1 Cross-test Evaluations

We note that the Weeds vocabulary from training and test sets slightly overlaps with the vocabularies of the rest of the datasets, thus the results on the classifiers trained on Weeds data may be influenced by the *lexical memorization* factor. Similarly, the classifiers trained on any other dataset when tested on the Weeds test set may be slightly influenced by the same factor.

6.8.1.1 Experiment 1: AUC ROC scores

Table 6.2 shows the AUC ROC scores on all the cross-test evaluations, which have a high confidence as the standard errors show. We can clearly see that classifiers trained on the Baroni dataset perform better than any other set of classifiers trained on any other dataset. Baroni classifiers do better in 5 out of the 6 test sets; thus, Baroni classifiers can do better than the rest of classifiers on any test set (except for the

Bless test set where Bless classifiers do better than Baroni classifiers.) This means that the Baroni dataset seems to be more useful for this ranking task, regardless of any threshold on the predictions, than any other dataset.

At a more fine-grained level of comparison, we can see that, indeed, the Baroni data seems to help classifiers to obtain better results. Half of the Baroni classifiers, across all the test sets, score in the [0.6-0.7] points interval and the rest of them score above 0.7 points. On the other hand, half of the Bless classifiers score just above 0.6 points, a rather poor generalization behavior. Two good results over 0.75 points are obtained, namely on the Baroni test set and on its own test set, though the last one is by far better than the former. In addition, Bless classifiers behave close to random on the Kotlerman test set. It seems that this dataset does not help the classifiers to generalize robustly to different distributions of data.

Regarding Levy classifiers, most of the scores obtained are below 0.7 points, and the best score (just above 0.7 points), is on the Baroni test set. This behavior found on Levy classifiers clearly indicates the inadequacy of the Levy dataset for the task. With a similar behavior to the Levy classifiers, we find that most of the Turney classifiers score below 0.65 points, and the highest score (0.68 points) is, again, on the Baroni test set. We can find the same pattern of behavior in the Weeds classifiers, where most of the of them score below 0.7 points and the best score (0.81 points) is on the Baroni data. Finally, we find that the Kotlerman classifiers provide the lowest results where most of the scores fall in the [0.5-0.6] points range.

We now analyze the AUC ROC scores from a slightly different perspective, namely from the point of view of the test sets. We observe that the highest scores are obtained on the Baroni test data, with values ranging from 0.65 to 0.91 points. Comparing with the closest test set, in terms of scores, we observe that results on the Bless test set ranges from 0.61 to 0.85 points. In average, the differences in scores obtained on the Baroni and the Bless test sets is 0.079 points, a notable difference. On the other hand, the test set on which classifiers obtained the lowest scores is on the Kotlerman data, where most of the results are below 0.6 points and the best score (0.61 points) is achieved by the Baroni classifiers.

Train \ Test	Baroni	Bless	Kotlerman	Levy	Turney	Weeds	max SE	Mean
Baroni	**0.916**	0.711	0.616	0.702	0.654	0.686	0.004	0.714
Bless	0.762	**0.850**	0.555	0.632	0.600	0.615	0.015	0.669
Kotlerman	0.653	0.612	**0.543**	0.566	0.581	0.544	0.020	0.583
Levy	0.716	0.611	0.592	**0.698**	0.569	0.533	0.017	0.619
Turney	0.686	0.646	0.547	0.595	**0.646**	0.520	0.015	0.606
Weeds	0.817	0.645	0.574	0.687	0.637	**0.675**	0.006	0.672

Table 6.2: Cross-test performance: Mean AUC ROC scores over 20 samples. Self-test score in bold. *Max SE*: maximum standard error of the mean across all means in a row. *Mean*: mean of the means in a row.

6.8.1.2 Experiment 2: Accuracy scores

In Table 6.3 we now observe the accuracy results of the different classifiers on all the adjusted test sets. In an overall picture, we see that most of the results, 75% to be more specific, are below 0.6 points, a low accuracy value. These results give us another perspective of the datasets. From this perspective, we observe that the Levy, Kotlerman, and Turney classifiers achieve scores mostly close to random across all the test sets, where the highest scores are 0.56 points (Kotlerman classifier), 0.52 points (Levy classifier) and 0.54 points (Turney classifier). A similar behavior is observed on the Bless classifiers where all the scores are near to random, and the only acceptable result (0.64 points) is achieved on its own test set. A slightly better picture can be observed on the Weeds classifiers where half of the scores are below 0.6 points and half of them are above such a score. Finally, the Baroni classifiers achieve, again, the best scores, where all of these scores, except for the score on the Kotlerman test set, are above 0.6 points, and the highest one –0.81 points– is achieved on its own test data. Comparing the results by columns (test data) in Table 6.3 we observe better scores on the Baroni test data, as in the previous experiment. All the classifiers achieve equal or better scores on Baroni test data than on its own test data, except for the Bless classifiers. Finally, we also observe that the classifiers with the highest confidence on the results are both the Levy and Baroni classifiers.

Train \ Test	Baroni	Bless	Kotlerman	Levy	Turney	Weeds	Max SE	Mean
Baroni	**0.812**	0.638	0.587	0.653	0.608	0.636	0.005	0.655
Bless	0.578	**0.642**	0.505	0.526	0.524	0.508	0.013	0.547
Kotlerman	0.563	0.546	**0.520**	0.524	0.528	0.528	0.014	0.534
Levy	0.521	0.510	0.507	**0.522**	0.509	0.496	0.004	0.510
Turney	0.546	0.534	0.518	0.540	**0.540**	0.479	0.016	0.526
Weeds	0.736	0.579	0.553	0.626	0.599	**0.600**	0.007	0.615

Table 6.3: Cross-test performance: Mean accuracy scores over 20 samples. Self-test score in bold. *Max SE*: maximum standard error of the mean across all means. *Mean*: mean of the means in a row.

6.8.2 Dataset Analysis

6.8.2.1 Experiment 1: Exploration of Datasets Along Generality and Similarity Patterns

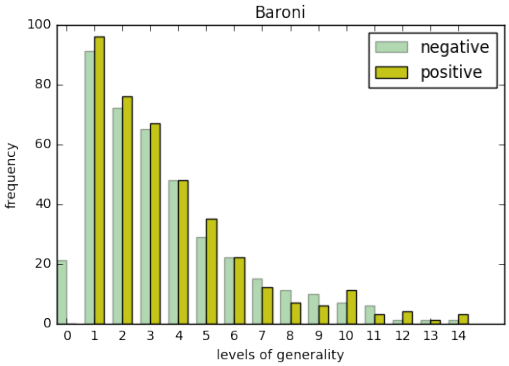
We plot how positive and negative instances distribute, along generality and similarity levels, for each training set of the datasets in Section 6.7.1.3. Figures 6.1, 6.2, 6.3, 6.4, 6.5, and 6.6 show such plots. We first look at the distribution of instances in the training sets, along generality levels, to see whether they comply with the two generality patterns described in Section 6.7.3.1. We note that across the generality levels $g \geq 1$ all the training sets are distributed as expected: The number of instances, positive and negative, is inversely proportional to the generality level; thus, at the highest levels we find the lowest concentration of instances while at the lowest levels we find the highest concentration. Though the positive instances in the Bless data are less compliant with this rule (Figure 6.2a); these instances seem to almost distribute uniformly up to level $g = 7$ and decrease in number afterwards. On the other hand, at the level $g = 0$, where we expect only negative instances, we find positive instances in three datasets, namely in the Kotlerman (Figure 6.3a), Levy (Figure 6.4a), and Turney (Figure 6.5a) datasets, though more markedly on the latter one where the ratio of positive vs. negative instances is the highest from these three datasets. These positive instances seem to be noisy instances. On the other hand, we observe that the Baroni (Figure 6.1a), Bless (Figure 6.2a), and Weeds

(Figure 6.6a) training sets do comply with our expectation by having only negative instances at the lowest generality level.

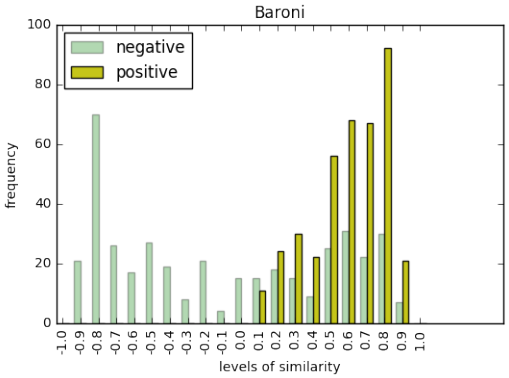
We now turn our attention to the expected similarity patterns. We find the first pattern in only two datasets, Baroni and Weeds. This pattern states that most of the positive instances are distributed proportionally to the level of similarity in the range $s \geq 0.1$; i.e., the higher the similarity level the more instances are concentrated, forming a sort of skewed-bell shape similar to that of the generality pattern but in the opposite direction (skewed to the right rather than to the left); however, the Weeds data seems to have some noisy positive instances populating the level $s = -1.0$. Closer to this pattern, but not fully compliant with, the Bless data has its positive instances distributed in the expected range, but they do not follow the expected skewed-bell shape. On the other hand, the Kotlerman, Levy, and Turney datasets have their positive instances distributed along all the similarity levels, including those levels reserved to negative instances, i.e. $s \leq 0.0$. We also observe that all the datasets, except for Weeds, comply with a pattern expected from the negative instances, namely that they are distributed along all similarity levels. Another pattern expected is the shape of the distribution of the negative instances; we find that only two datasets, the Baroni and Levy, are the closest to a sort of double-bell shape peaked at both the lowest and the highest levels of similarity. The rest of the datasets have their negative instances distributed in a less clear manner.

6.8.2.2 Experiment 2: Creation of a New Dataset by Leveraging Generality and Similarity Patterns

We experimentally test our hypothesis that the characterization of hypernymy along *generality* and *similarity* patterns is useful in the construction of a hypernymy dataset. To do so, we create a new dataset that mimics such patterns from the most useful hypernymy dataset. According to the evidence derived from our cross-test evaluations (Sections 6.8.1.1 and 6.8.1.2) and data analysis (Section 6.8.2.1), we find that the Baroni dataset is both the most useful to extract hypernym relations from word embeddings and the one that complies with all the expected patterns. Using the same criterion, we find the Kotlerman dataset to be the least useful for

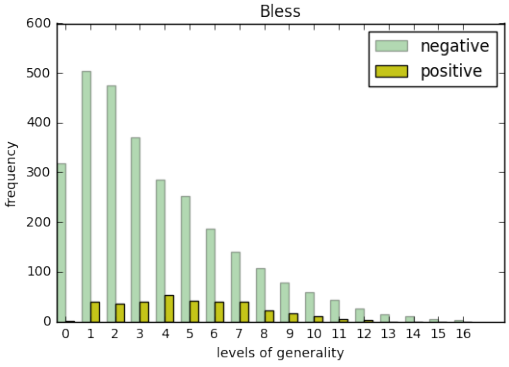


(a) Generality Patterns

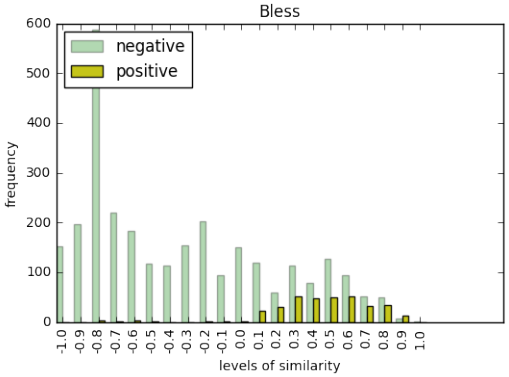


(b) Similarity Patterns

Figure 6.1: Distribution of positive and negative instances of the Baroni training set along generality and similarity levels.

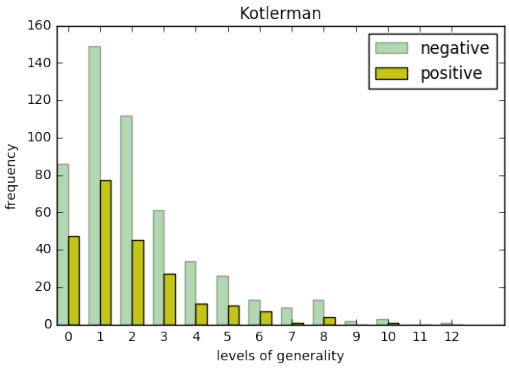


(a) Generality Patterns

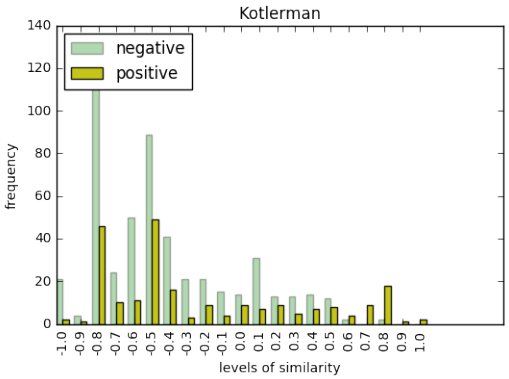


(b) Similarity Patterns

Figure 6.2: Distribution of positive and negative instances of the Bless training set along generality and similarity levels.

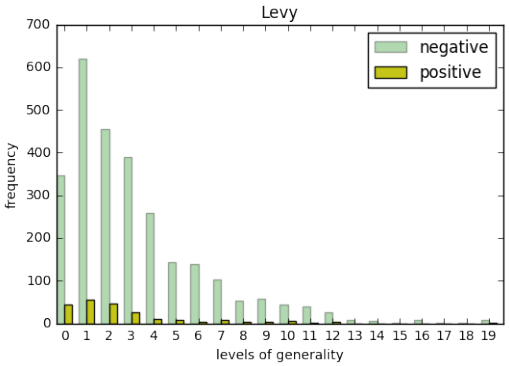


(a) Generality Patterns

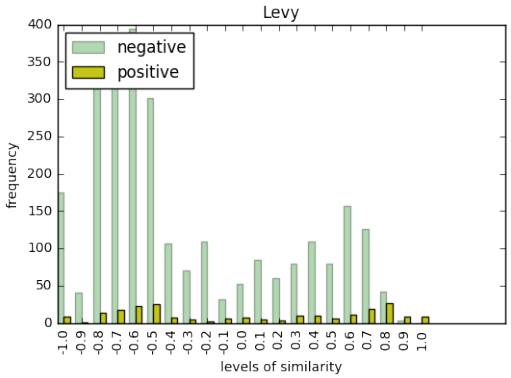


(b) Similarity Patterns

Figure 6.3: Distribution of positive and negative instances of the Kotlerman training set along generality and similarity levels.

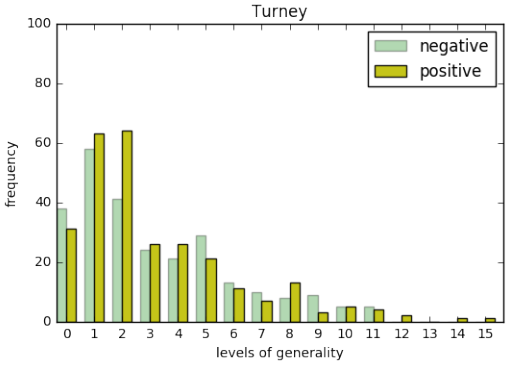


(a) Generality Patterns

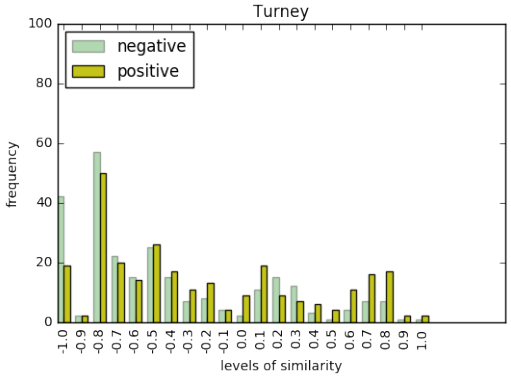


(b) Similarity Patterns

Figure 6.4: Distribution of positive and negative instances of the Levy training set along generality and similarity levels.

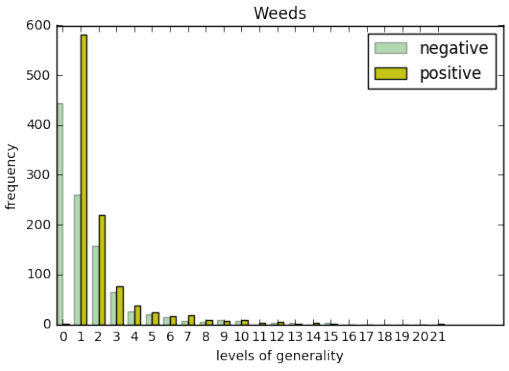


(a) Generality Patterns

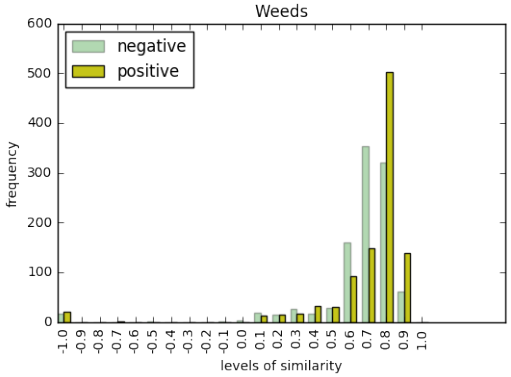


(b) Similarity Patterns

Figure 6.5: Distribution of positive and negative instances of the Turney training set along generality and similarity levels.



(a) Generality Patterns



(b) Similarity Patterns

Figure 6.6: Distribution of positive and negative instances of the Weeds training set along generality and similarity levels.

hypernymy extraction. As we explained in Section 6.7.3.2, we create rules in order to sample instances from the union of all the training sets from Section 6.7.1.3. Examples of the rules we use in order to accept an instance $h = (c_i, c_j, label)$ to populate the new dataset are those shown in Examples 6.15 and 6.16. Similar rules are used in order to create the dataset that is distributed as the Kotlerman dataset.

(6.15) IF generality value $g(c_i, c_j) = 0$ AND $label \neq True$ AND number of positive instances = 0 in generality level $g = 0$ AND number of negative instances ≤ 11 in generality level $g = 0$ THEN accept instance sampled

(6.16) IF generality value $g(c_i, c_j) = 1$ AND ratio of number of positive instances vs. number of negative instances > 1.06 in generality level $g = 1$ AND ratio of number of positive instances vs. number of negative instances < 1.10 in generality level $g = 1$ AND total number of instances < 97 in generality level $g = 1$ AND total number of instances in generality level $g = 1$ is greater than the total number of instances at any other generality level THEN accept instance sampled

Results of the classifiers trained on the new dataset and tested across all adjusted test sets are presented in Tables 6.4 and 6.5. We call this dataset *B2* which stands for *Baroni version 2* since it mimics the patterns from the Baroni data. We compare it against the baseline dataset *No-rules* and *K2*; the first name reflects the fact that we did not use any rule to accept instances to populate that dataset, while the second name stands for *Kotlerman version 2*. We observe that *B2* classifiers achieve better, or equal, AUC ROC and accuracy scores than the two baselines in all cases. Furthermore, the standard error in the results of *B2* classifiers are the lowest ones. We also observe a similar trend if we compare the two baselines; *No-rules* classifiers achieve better, or equal, scores than *K2* classifiers in all cases except for one test set, the Weeds test, in terms of accuracy. From this evidence we can see that by virtue of arranging the data according to generality and similarity patterns we can either improve or worsen the ability of classifiers to extract hypernym relations from word embeddings.

In Table 6.6 we can observe the distribution of instances of the three new datasets according to the original datasets they were sampled from. We observe a clear tendency among the three sets of data; the proportion of instances coming from a particular dataset increases or decreases along with the scores of the classifiers trained on those instances. For example, *B2*, the training set that leads to the best scores, contains more instances from the Baroni data (13.8% of its sample) than both *No-rules* (9.2%) and *K2* (8.8%). A similar figure is observed across the rest of the instances, being the instances from the Baroni and Bless datasets more numerous in *B2* than in the baseline datasets; and instances from the Kotlerman, Levy, Turney, and Weeds datasets are more numerous in the least useful dataset, namely *K2*, than in *B2*. Thus, it seems that the rules we used to populate *B2* tend to accept more Baroni and Bless data while excluding instances from the other datasets.

Train \ Test	Baroni	Bless	Kotlerman	Levy	Turney	Weeds	Max SE	Mean
B2	0.879	0.734	0.618	0.692	0.641	0.672	0.011	0.706
No-rules	0.859	0.725	0.603	0.688	0.641	0.655	0.013	0.695
K2	0.852	0.724	0.597	0.685	0.639	0.654	0.015	0.691

Table 6.4: New hypernymy datasets: Mean AUC ROC scores over 20 samples. *Max SE*: maximum standard error of the mean across all means in a row. *Mean*: mean of the means in a row.

Train \ Test	Baroni	Bless	Kotlerman	Levy	Turney	Weeds	Max SE	Mean
B2	0.794	0.664	0.580	0.644	0.596	0.629	0.011	0.651
No-rules	0.775	0.655	0.566	0.641	0.596	0.598	0.013	0.638
K2	0.763	0.645	0.561	0.639	0.596	0.615	0.015	0.636

Table 6.5: New hypernymy datasets: Mean accuracy scores over 20 samples. *Max SE*: maximum standard error of the mean across all means in a row. *Mean*: mean of the means in a row.

6.9 Discussions and Conclusions

We first answer our research questions previously posed and then we elaborate on both the answers to the questions and the findings from our experiments.

Our cross-test evaluations suggest that *GloVe* word embeddings capture hypernymy information, despite the fact that the training function of *GloVe* does not

Train \ Source	Baroni	Bless	Kotlerman	Levy	Turney	Weeds
B2	13.814	29.085	6.600	20.811	7.382	22.305
No-rules	9.289	26.855	7.907	21.302	7.881	26.763
K2	8.839	21.757	10.957	23.204	8.971	26.269

Table 6.6: Distribution of instances (mean percentage across 20 samples) in the *B2*, *No-rules*, and *K2* datasets according to the source of origin.

include an explicit term dedicate to learn hypernymy. Furthermore, we found that the selection of a hypernymy dataset is a crucial step in the experimental setup to extract hypernymy information. According to our evaluations and data analysis, we conclude that out of the six datasets analyzed one is substantially more useful than the rest for extracting hypernymy from word embeddings –the Baroni dataset. To do these evaluations, we controlled for two confounding factors in the datasets, namely sample size and imbalance. Furthermore, we hypothesized that part of the usefulness of a dataset is due to its design: Creating a dataset by following a background theory of the phenomenon under study helps to discard data not representative of the phenomenon. Further experiments show evidence that, indeed, a dataset built following a characterization of hypernymy is more useful for the task than a dataset that is not. According to our analysis we thus conclude that the Baroni data is useful for extracting hypernymy from word embeddings, and part of its usefulness is due to its construction design which follows two patterns relevant to hypernymy, namely *generality* and *similarity*. These results may help other people in the NLP community to better exploit the word embeddings in tasks where both hypernymy is required and knowledge transfer, in the form of word embeddings, is possible.

6.9.1 On the Cross-Test Evaluation of Hypernymy Datasets From the Literature

In order to fairly compare the hypernymy datasets we adjusted them to avoid two possible confounding factors influencing the results in our cross-test evaluations. We controlled for sample size and imbalance, in the case of the training sets, and

we did it by bootstrapping 20 training sets from each of the original training sets. Each new training set thus contained the same number of instances as the other bootstrapped samples where the number of positive instances was the same as that of negative instances. Then we trained one classifier for each new training set and averaged the scores across the 20 classifiers. This setting allowed us to obtain confidence values. In the case of test sets, we controlled only for imbalance since there was no need to control for sample size.

In our cross-test experiments, we found that classifiers trained on the Baroni data achieve, on average, the highest scores across all the test sets. AUC ROC and accuracy scores obtained by these classifiers are consistent with each other. The ability of Baroni classifiers to extract hypernymy is always above the chance level (0.5 points) in both scenarios, AUC ROC and accuracy scores. In contrast, classifiers trained on Bless, Levy, and Turney data show some AUC ROC scores which disagree with accuracy scores. Using the former metric, the results portray classifiers with an ability of ranking positive instances above chance level, whereas accuracy results show the opposite picture; most of the accuracy results show that the classifiers are practically *tossing a coin* when predicting. This discrepancy between AUC ROC and accuracy scores, not seen in the rest of the datasets, is likely due to the quality of the validation data where we optimized the thresholds; this means that the validation sets of the *discrepant* datasets are not useful. The Baroni data seems to be the only dataset with representative data of hypernymy well distributed across training, validation, and test sets. All of this cross-test results show low standard error of the mean across all classifiers, which provide us with a high confidence in the results.

Comparing the metrics we used, AUC ROC and accuracy, with that used in previous work, F_1, we observe some differences. First, previous work reports results using a single scoring metric, which provides scores only for the minority class (positive instances) and requires a threshold. In contrast, we provide two metrics, widely used in other binary classification problems, which provide scores for both classes, minority and majority, as well as scores of the ranking behavior of the

classifier across several thresholds. Second, F_1 is not as intuitive to interpret as accuracy which provides easy-to-interpret results.

6.9.2 On the Analysis of Hypernymy Datasets Along *Generality* and *Similarity* Patterns

We hypothesized that a hypernymy dataset encoding *generality* and *similarity* patterns would be more consistent and thus more helpful for the classifying hypernymy. We tested our hypothesis via two experiments. First, we analyzed the specific generality and similarity patterns encoded in each of the datasets under study; we compared these patterns against an expected structure in a hypernymy dataset. We derived this theoretical structure based on our background theory of hypernymy together with a thought experiment of sampling instances from a hypothetical taxonomy. In our experiment, we found that most of the datasets have their instances distributed as expected across generality levels $g \geq 1$, but only half of the datasets comply with our expectation at generality level $g = 0$. This seems to indicate that the sampling procedures used to obtain the data were appropriate –uniformly at random– but the positive instances found at level $g = 0$ may be noisy or incorrectly labeled instances, i.e. false hypernym relations. As for similarity patterns, we found that most of the datasets fulfilled one of our expected patterns, namely they have their negative instances distributed along all similarity levels. However, only the Baroni dataset complies with the rest of the patterns, namely the shape of the distribution of negative and positive instances. In contrary, datasets such as Kotlerman, Levy, and Turney have their positive instances distributed in a totally unexpected manner; these instances are spread along all the similarity levels, including those levels exclusive for negative instances where the similarity between two concepts indicates a lack of relatedness between such concepts. Overall, the only dataset that complies with our expected structure is the Baroni dataset.

We further hypothesized that the theoretical structure that we proposed would play a role in the usefulness of a hypernymy dataset to allow a classifier to predict hypernym relations. We tested this hypothesis via a second experiment where we created new datasets in the light of our above findings. Each dataset was sam-

pled from the same pool of instances, but using a different sampler. One sampler followed certain sampling rules in order to distribute the instances as similar as those in the Baroni dataset, the most consistent and useful dataset as our cross-test evaluations showed. Another sampler followed sampling rules with the objective of copying the distribution of the least useful and consistent dataset, namely the Kotlerman dataset. The last sampler selected instances from the pool uniformly at random, i.e. it did not follow any specific rule to imitate any distribution. We obtained 20 training samples from each of the three samplers by bootstrap. Each training sample was used to learn a classifier which in turn was evaluated on all the test sets previously used in the cross-test experiments. We corroborated our hypothesis; we found that the classifiers that achieved better AUC ROC and accuracy results were those trained with the data distributed as the Baroni data. On the other hand, the classifiers with the lowest scores were those using training samples distributed as the Kotlerman data. The classifiers using data obtained with no prior assumptions achieved scores in between the other two sets of classifiers' scores. This indicates that structuring a hypernymy dataset according to a certain structure along generality and similarity patterns may either leverage (as the training samples following the Baroni structure did) or worsen (as the training samples following the Kotlerman structure did) the abilities of the classifiers to extract hypernymy from word embeddings.

6.9.3 On Limitations in This Work

We also found both limitations in our work and possible factors that may have affected our results. A possible factor influencing our dataset analysis is the use of a specific taxonomy (WordNet) in order to measure generality and similarity levels. We are not clear to what extent having used any other taxonomy would have changed our outcomes, since each taxonomy reveals the creator's view of the world. Other factors, at a finer-grained level of analysis, unconsidered in this work are those related to the specific vocabularies used by each dataset. For example, it may be possible that the semantics of monosemous concepts are better captured by the word embeddings than that of polysemous concepts, and thus the former are easier

to predict in a hypernymy relation. On the other hand, a limitation of our work is the fact that we left out of the study more recent hypernymy datasets, such as *HyperLex* (Vulić et al., 2017). We consider that all these factors and limitations may well be studied in future work.

Chapter 7

Conclusions

In this thesis, we studied five NLP ReLe systems, namely, *Model F*, *ESIM*, *DAM*, *CE*, and *GloVe*. We focused on the study of three aspects, namely *interpretability*, *robustness*, and *abilities learned*; we advocated for studying these aspects in order to better understand a) how a system makes a prediction, b) how robust is a system to challenging circumstances and what factors influence its predictive behavior, and c) to know whether a system has learned a specific linguistic ability. Doing so represented a challenge since these systems are a type of *black-box*, i.e. they are not easy to interpret by a human. Thus, we used three types of analysis that treat the systems as black-boxes, namely *representation-level analysis*, *behavior analysis*, and *internal analysis*. We divided our thesis into three case studies where each study was dedicated to the one of the three aspects.

Furthermore, across our three case studies, we show pieces of evidence in favor and against previous ideas in the NLP community towards certain capacities and abilities of ReLe systems. According to our analyses, ReLe systems seem to learn correlations of features from the datasets. Moreover, ReLe systems also seem to learn biases from the data, and to use them to predict. Furthermore, ReLe systems seem to be insensitive to some changes in the input space, specially changes that generate instances very similar to those from the training data. Also, ReLe systems seem to struggle with instances that are significantly different from those in the training data. Nevertheless, ReLe systems seem to be able to capture some type of semantic information in their parameters. Thus, our analyses bring into question

previous ideas about some capacities and abilities of ReLe systems, such as their ability to generalize, their capacity of understanding language, and their ability for learning semantic information. More concretely, it is not clear how well ReLe systems are able to generalize to any novel instance; as we mentioned before, they seem to predict based on heuristics in the form of feature correlations and biases found in the data, probably not the best strategy for generalizing. Also, it is not clear to what extent ReLe systems use a language-understanding capacity to classify instances in the form of natural language sentences; our results show that NLI ReLe systems based their predictions on confounding factors, such as a bias from the data, and when ReLe systems do not rely on these factors, they significantly lose accuracy. Nevertheless, our analyses show that ReLe systems seem to encode hypernymy, a type of semantic information, as previously hypothesized in the NLP community; however, it is not clear to what extent hypernymy is encoded due to the capability of ReLe systems to encode the semantics (meanings) of concepts and information about their relations, or due to learning statistical patterns from the data that work as an approximation of hypernymy information. Overall, we show the importance of studying ReLe systems to provide evidence towards *demystifying* ascribed capabilities to them and to better understand their actual abilities.

In the following sections we summarize each of our case studies, we answer our research questions posed in Chapter 1, we then provide a summary of our main contributions, and finally, we describe limitations and future work.

7.1 A summary of Our Three Case Studies

7.1.1 First Case Study: Explaining Predictions of *Model F*

In this study (Chapter 4), we analyzed *Model F*, a matrix factorization system for the task of knowledge base population. We studied the interpretability of this system via a representation-level analysis, i.e. we investigated how to obtain an explanation of how *Model F* makes a decision. To do so, we borrowed methodology from previous work in interpretability; we used a *pedagogical* approach where we treated *Model F* as an oracle in order to label input instances. With these instances we created a

dataset exposing the predictive behavior of *Model F*; then, we trained interpretable proxy models with this dataset so they could learn how to mimic the behavior of *Model F*. We note that these proxy models where easy to interpret, i.e. we were able to obtain an explanation of how a prediction was made by looking at their structure and parameters. We used two proxy models from the literature, namely logic rules and decision trees, and a new proxy model that we proposed, namely a tree-structured Bayesian network. We found out that the two proxy models from the literature were not able to fully capture the predictive behavior of *Model F*, being logic rules the least successful model. A possible reason for these results may be the heuristics that we used to learn the logic rules; we chose a restricted form of Horn rule where only one predicate is present in the body of the rule. Another reason may be due to the nature of the models; logic rules are a symbolic model, and thus trying to capture the probabilistic, ranking behavior of *Model F* seems to be difficult; and similarly, decision trees have been shown to have difficulty in handling ranking tasks. On the other hand, our proposed model, a Bayesian network tree (BN tree), was able to faithfully capture the predictive behavior of *Model F*. Furthermore, we showed how our BN tree had a very similar ranking behavior to *Model F* when generalizing to test instances.

This evidence shows that the BN tree is an equivalent of *Model F*. The behavior of the BN tree is functionally similar to that of the ReLe system. Based on this evidence and the fact that a BN tree is easy to interpret, we provided explanations of incorrect predictions made by *Model F* through the BN tree. By looking at the tree structure we were able to spot *spurious* edges that connected the output with the input variables; i.e., we were able to graphically see how *Model F* incorrectly related the input to the output; this explanation was shown as a multi-step decision process that *Model F* did in order to arrive at the observed output given the input.

Furthermore, we claimed that this multi-step decision process, given by the BN tree, accounts as an explanation of the inner workings of *Model F*. We described how learning a proxy model that explains the predictive behavior of a black-box system accounts as a form of representation-level analysis from cognitive science. On

the one hand, a proxy model relates the input to a system to the response of the system in an algorithmic way, i.e. the proxy model proposes a plausible, faithful logic of how the system arrives to the observed output, given the inputs, without having to inspect the internal mechanisms of the system. On the other hand, a representation-level analysis seeks to explain how an information-processing system processes information regardless of the mechanisms that give rise to this process, and thus it is based on input-response observations of a person; i.e., a representation-level analysis provides a plausible logic of how a system arrives to an observed output given an input.

7.1.2 Second Case Study: Evaluating Robustness of *ESIM*, *DAM*, and *CE*

In this study (Chapter 5), we analyzed three ReLe systems for the task of natural language inference, namely *ESIM*, *DAM*, and *CE*, being the first one a state-of-the-art system on the SNLI dataset. We aimed to evaluate their robustness to challenging instances that we generated when we made a simple transformation to input instances, namely a swap of word pairs. For example, given the following *entailment* instance we swap the word pair *(footbridge, bridge)*:

(7.1) p : A little girl hugs her brother on a footbridge in a forest.
h : A pair of siblings are on a bridge.

And thus we generate a transformed instance of class *neutral*:

(7.2) p : A little girl hugs her brother on a bridge in a forest.
h : A pair of siblings are on a footbridge.

Then, we aimed to answer the question: To what extent are the systems robust to this transformation? Given that it was difficult to control for certain confounding factors, we statistically analyzed them to elucidate to what extent they played a role in the predictive behavior of the systems; we named these factors *insensitivity*, *unseen word pairs*, and *polarity*.

Insensitivity is an internal factor to the systems; it refers to the ability of the systems to recognize that an input instance has changed. Thus, we measure to what

extent the systems correctly react to our transformation. In the above examples, if the systems were insensitive to the swapping of the word pair *(footbridge, bridge)* then they would classify both instances as type *entailment* since they would not recognize that we swapped a word pair and thus generated a new instance of class *neutral*. *Unseen word pairs* are those word pairs that the systems did not see at training time and thus may influence the behavior of the systems; for example, the swapped word pair from the example above, *(bridge, footbridge)* maybe was never seen in any training instance, and thus the systems may not know how these two words interact with each other affecting their predictive behavior. Finally, *Polarity* is a term we use to label word pairs into a certain class depending on the class of instances these word pairs were most frequent in training data; for example, we associate the word pair *(footbridge, bridge)* with the label *entailment* because this pair was mostly seen in *entailment* instances.

Thus, following methodology from the behavioral science, we statistically correlate these factors with the response of the systems when we feed them with the challenging instances that we generated in order to elucidate whether these factors influence the systems' behavior. According to our results, we found that these factors do play a role in the predictive behavior of the three systems. *Insensitivity* helps the systems to obtain high accuracy on challenging instances that have the same class label as the instances they were generated from; in other words, the systems' lack of sensitivity to recognize that we swapped a word pair helps them to predict the same class label in both cases, the instance they were familiar with and the transformed (challenging) instance. When the transformed instances have a different label from the instances they were generated from, then we can notice how the systems significantly drop accuracy, according to our statistical tests. *Unseen word pairs* is an interesting factor in the sense that depending on the semantic relation of the two words in the pair is how robust are the systems. We found out that unseen antonym pairs affect the systems' performance, i.e. the systems did not learn antonymy; on the other hand, the systems are robust to unseen hypernym or hyponym pairs; for example, if the systems learned that the pair *(footbridge, bridge)*

is a hyponym pair, then when they encounter the unseen pair *(bridge, footbridge)*, they correctly recognize that now the words are in a hypernym relation. This finding goes in hand with our main finding in our third case study, namely that a ReLe system is able to learn hypernymy. *Polarity* of word pairs seems to be a factor also influencing the systems' predictions, as our statistical analyses show; the three systems learned to predict the label of an instance based on the polarity of a word pair inside the instance; for example, if the word pair *(bridge, footbridge)* had polarity *neutral* in the instance in Example 7.2, then the systems likely would predict *neutral* as the class label for this instance, regardless of the rest of the words in the instance.

These findings not only apply on transformed instances that we generated from original instances contained in the SNLI dataset, but also on other type of instances that we generated which we call *ex situ*. This type of instances keep fixed the words surrounding a word pair; for example, the instances in Examples 7.1 and 7.2 would be transformed into the following instances:

(7.3) p : A little girl hugs her brother on a footbridge in a forest.
h : A little girl hugs her brother on a bridge in a forest.

(7.4) p : A little girl hugs her brother on a bridge in a forest.
h : A little girl hugs her brother on a footbridge in a forest.

Where Example 7.1 was transformed into its *ex situ* equivalent shown in Example 7.3, and similarly, Example 7.2 was transformed into Example 7.4. In this way, we say that we have two types of instances, namely *in situ* and *ex situ*. Analysis on *ex situ* instances allowed us to study the effect of swapping a word pair when the surrounding words were fixed. Interestingly, the systems' behavior on *ex situ* instances of class *contradiction* is similar to that on *in situ* instances of the same class; the systems are influenced by the same factors in similar ways and the accuracy scores are comparable (except for the *CE* system that significantly drops accuracy on *ex situ* instances.) However, when when the instances are class *entailment* or *neutral*, *ESIM* and *CE* significantly drop accuracy on *ex situ* instances when compared to *in situ* instances, while *DAM* is able to cope equally with both types. In addition, in these *ex situ* instances, all the systems are also influenced by

the confounding factors mentioned above.

In conclusion, we found that none of the systems is fully robust to our transformation on input instances, being the three systems highly influenced by the factors *insensitivity*, *polarity*, and *unseen word pairs*. Our statistical analyses, borrowed form the behavioral science, provide us with *interval validity*; i.e., we have high statistical confidence that the response observed from the systems is indeed associated with the factors analyzed.

7.1.3 Third Case Study: Extracting Hypernymy From *GloVe*

In this study (Chapter 6), we investigated to what extent we can extract hypernymy, a semantic relation between two words, from the parameters of *GloVe*, a ReLe system dedicated to learn word embeddings. We categorized this study as an internal analysis due to the similarity to work from neuroscience where information is extracted from neural activity (in the form of fMRI) of a person using a classifier that reads this representation of the brain's activity and predicts whether specific information is contained in there. In a similar setup, we aim to elucidate to what extent we can extract hypernymy from *GloVe* word embeddings using a classifier trained on this vector representation. Previous work tried to do this, but the results obtained were inconclusive due to a lack of a proper control in the experimental setup, where the key factor to control for is the choice of hypernymy dataset. Thus, we proposed an appropriate experimental setup where we studied how the choice of a hypernymy dataset influenced the behavior of the classifiers trained to extract hypernymy from word embeddings.

Our analyses showed that from six datasets analyzed only one is consistent, and thus useful, for recovering hypernymy from *GloVe* embeddings, namely the Baroni dataset. These conclusion is based on two experimental analyzes: A cross-test evaluation and a structure analysis. On the one hand, in our cross-test evaluation we compared which dataset allows classifiers to perform better on the test sets of the six different datasets. To do this, we controlled for two possible confounding that may influence the classifiers, namely the sample size of the datasets and imbalance between positive and negative instances. Thus, we generated bootstrapped datasets

that were equal in size and balanced. Under this controlled scenario, we found that the Baroni dataset allowed classifiers to perform better than any other dataset across the different test sets.

On the other hand, in our structure analysis, which we based on psychology theories of concept categorization, we hypothesized that the usefulness of a hypernymy dataset is due to two properties fundamental for hypernymy: *Generality* and *similarity*. Two concepts can be classified to hold under a hypernymy relationship if one is more abstract (general) than the other and both are similar to each other. For example, we say that a *dog* is a type of *vertebrate* because the latter concept is a more abstract concept; this can be shown in how all the features of *vertebrate* are inherited by *dog*, but not the other way around (not all *vertebrates* bark as *dogs* do.) Also, both concepts are similar to each other in the sense that both are classified into the same taxonomic sub-tree, namely the *animal* taxonomy; thus both concepts share features proper of any *animal*. We operationalized the concepts of *generality* and *similarity*, in order to measure them on hypernymy datasets, based on metrics from WordNet. We defined *generality* of two concepts A and B as the difference in the number of levels in WordNet from A to B with respect to the root of the taxonomy. We defined *similarity* of two concepts with the Wu-Palmer similarity function which provides a measure of how similar are two concepts based also on the difference in levels within the taxonomy.

In this way, we proposed an expected structure to be found in an useful hypernymy dataset based on our conceptualization of hypernymy along generality and similarity properties. Then, we analyzed the hypernymy datasets from the literature in order to find out which dataset fulfills our expected structure. Interestingly, only one dataset fulfilled our proposed structure, namely the Baroni dataset. In this dataset, the positive and the negative instances are distributed along generality and similarity levels as we expected; for example, positive instances, such as *(dog, animal)* have generality and similarity values that fall in a certain range of values; more concretely, they are distributed in the generality levels of $g \geq 1$ (the level $g = 0$ is only for co-hyponyms, a type of negative instance) and we expect them to

be distributed following a type of left-skewed bell. Similarly, we expect positive instances to be distributed in the similarity levels of $s \geq 0.1$ forming a right-skewed bell. We also proposed similar patterns for negative instances across both generality and similarity levels. To further prove our hypothesis, we created new datasets using instances from the six datasets analyzed; we do so by enforcing the new datasets to hold the properties of generality and similarity that we expect to find in an useful hypernymy dataset. When we used these new datasets, we found improvements in accuracy and AUC ROC scores when extracting hypernymy, as opposed to when we create datasets that do not hold the two properties mentioned before.

Overall, our studies found that by using the most consistent dataset, in a supervised approach, we can accurately predict whether two *GloVe* embeddings hold under a hypernym relationship. Thus, we conclude that, indeed, it is possible to extract hypernymy information from *GloVe* word embeddings, despite that the loss function of this ReLe system did not explicitly account for learning such linguistic phenomena.

7.2 Answering Our Research Questions

Throughout this thesis, we aimed to study and understand the behavior and the abilities learned of specific ReLe systems. To do so, we proposed to focus on both specific aspects to study (interpretability, robustness, and abilities learned) and specific types of analyses to carry out the studies (representation-level, behavioral, and internal analyses.) We proposed to look into other disciplines to borrow certain instruments, such as research questions, methodologies, motivation, or analyses. In particular, we focused on three disciplines, namely cognitive science, behavioral science, and neuroscience.

We gave as an example the following questions addressed by each of these disciplines: What is a cognitive model that explains how the information is processed by the long-term memory? How does the choice of payment instrument influence the spending behavior of people? Is it possible to extract information from the brain activity of people? The study of each of these questions allows science to obtain ex-

planations of the target phenomena and, thus, better understand particular aspects of the subject under study. Then, we asked the questions: Can we study ReLe systems in a similar way? Can we provide an explanation, in the spirit of cognitive science, of the behavior of a representation learning system? Can we know what external and internal factors influence the predictive behavior of a system? Can we extract information from the internal components of a system? We showed throughout this thesis that the answer is affirmative.

In order to addressed the above questions, we borrowed different instruments from different disciplines, mainly from previous work in the machine learning and natural language processing communities, but also from the disciplines listed above. More concretely, in our first case study, we borrowed methodology from previous work on interpretability though we borrowed motivation and a research question from cognitive science, and we draw a parallel between interpretability and representation-level analysis.[1] We showed how an interpretable proxy model accounts may be viewed as an equivalent of a representation-level analysis of a black-box system. Thus, we tied together these two types of analyses. This allowed us to categorize a proxy model as a type of explanation of the inner working of a ReLe system aligned to analyses from the cognitive science. In our second study, we borrowed motivation from the natural language processing community while we took research questions and methodology from the behavioral science;[2] in this way, we were able to evaluate robustness of ReLe systems by controlling for confounding factors while we statistically analyzed what other factors influenced the systems' response. Finally, in our third case study, we also draw a parallel of how the objectives and methodology of information decoding in neuroscience aligns to those in the field of hypernymy prediction, though we mainly borrowed all the experimental setup from the latter field.[3]

[1] We do not claim as our contribution any method for doing interpretability analyses; we claim as our contribution comparing how an interpretability analysis is similar to a representation-level analysis from cognitive science.

[2] To clarify, we claim as our contribution being the first to conduct analysis of ReLe systems using methods from behavioral science.

[3] As in our first study case, we do not claim as our contribution any method for extracting hypernymy from embeddings; we claim as our contribution comparing how this analysis is similar to

We also asked in Section 1 what types of explanations can we then obtain from our studies. We obtained three types of explanations. The first one, an explanation, at the representation-level, of how the matrix factorization from Chapter 4 works; this type of explanation provides us with the logic of the decision process that the ReLe system follows in order to produce an output; i.e. it describes how the ReLe system processes the input information in order to produce an output. Although this explanation describes the inner working of the target system, it does not need to open the system and study its internal machinery, rather it only needs input-output observations of the system. Based on this explanation we were able to understand predictions from the ReLe system by looking, in a graphical way, at a multi-step decision process that the target system performs. Our second type of explanation is in the form of a set of statistics along with their description. More concretely, we provided an explanation of how ReLe systems were influenced by certain external and internal factors when predicting an output. The form of this explanation is that of statistical associations between a factor and the system's response; in this way, we were able to explain how the SNLI systems from Chapter 5 were influenced by three target factors and how these factors affected their robustness. An advantage of this type of explanation is that they provide us with internal validity; i.e., we can guarantee with some confidence that changes in the response of the target system are due to the factor being analyzed. Our third explanation, in our third case study in Chapter 6, is not about understanding the ReLe system, but rather understanding how hypernymy datasets should be structured in order to be useful. We showed that by characterizing hypernymy along *generality* and *similarity* properties we can propose a structure of how to sample and distribute positive and negative instances. This explanation not only allowed us to better understand how only one hypernymy dataset from the literature was useful for extracting hypernymy from word embeddings, but also it allowed us to create better datasets.

In summary, we worked towards better understanding ReLe systems by nurturing our work from other disciplines whose main objective is to provide explanations

analysis done in the neuroscience for decoding semantic information from brain activity.

of diverse phenomena occurring in different subjects. We provided qualitative analysis and different types of explanations which we hope the community will use in order to both further understand these type of systems and to improve the next generation of ReLe systems.

7.3 Summary of Major Contributions

We list the major contributions of this thesis in a chapter by chapter basis.

1. Chapter 4

    - We studied a new type of ReLe system, namely the matrix factorization system *Model F*, which was trained with a ranking loss function rather than with a classification loss function as most of the systems from the literature.
    - We provided a new interpretable proxy model, namely a Bayesian network tree; according to our evidence, this model faithfully captures the predictive behavior of *Model F*. In addition, we show that popular proxy models from the literature, namely logic rules and decision trees, were not able to capture the behavior of *Model F*.
    - We also proposed a new evaluation metric of fidelity, namely precision-recall curves, since those from the literature, F_1 and accuracy, were used for classifiers.
    - We showed that our interpretable proxy model is equivalent to a representation-level analysis, which allows us to understand how *Model F* makes a prediction. (We clarify, however, that interpretability analysis is not our contribution; we contribute with comparing this analysis with representation-level analysis to show similarities from both.)
    - By showing that our Bayesian network's behavior is functionally similar to that of the MF system, we threw some light of a plausible way the MF operates, namely by learning correlations in the data.

- We published two papers, an AAAI spring symposium paper and a NIPS workshop paper, based on the work of this chapter.

2. Chapter 5

   - We evaluated the robustness of three SNLI ReLe systems, namely *ESIM* (a state-of-the-art system), *DAM*, and *CE*. This evaluation allowed us to see that none of these systems is as robust as previously thought.
   - We used methodology from the behavioral science in order to control for confounding factors and to investigate how certain factors influence in the predictive behavior of these systems, sometimes giving a misleading picture of robustness. We believe to be the first to borrow methods from the behavioral science to analyze ReLe systems at the stimulus-response level.
   - We proposed a transformation on input data in order to do the evaluation of robustness. This transformation allowed us to generate challenging test sets that we will release to the community.
   - Furthermore, we provided explanations in the form of statistical correlations that allowed us to see, with high statistical confidence, how the factors under study played a role in the systems' response.
   - Moreover, we found that there are behavioral patterns across the three systems; they are affected in the same way by the target factors.
   - We provide a piece of evidence towards NLI ReLe systems classifying instances based on confounding factors, bringing into question the idea of ReLe systems predicting based on understanding the semantics of the instances. However, we acknowledge that our results are only a piece of evidence, and in order to conclusively prove any hypothesis about the capabilities of ReLe systems, dedicated studies should be carried out.
   - We got accepted a long paper at the North American Chapter of the Association for Computational Linguistics 2018.

3. Chapter 6

    - We provided evidence towards *GloVe* embeddings capturing hypernymy, a type of semantic relation between two concepts.
    - We also provided a comparison of six hypernymy datasets from the literature. To do this comparison, we controlled for confounding factors and then we measured the usefulness of these datasets for training classifiers to extract hypernymy. Furthermore, we analyzed the structure of these datasets to see which ones comply with a structure that we proposed based on work on psychology. We found that only one dataset out of the six datasets is useful for hypernymy extraction.
    - Moreover, based on our proposed structure of a hypernymy dataset, we were able to create new datasets that are also useful for extracting hypernymy.
    - Also, we proposed to evaluate classifiers with accuracy and AUC ROC metrics, instead of F_1 as in the literature, which provided a better insight into the classifiers behavior.
    - Even though we borrowed the methodology we use to extract hypernymy from the NLP literature, we contribute with drawing a parallel between this type of analysis from NLP with that from neuroscience (brain decoding), and we show how both types of analysis are similar in both objectives and methods.
    - We published a short paper on the European Chapter of the Association for Computational Linguistics 2017 based on this chapter.

7.4 Limitations and Future Work

We first present limitations shared across our three studies, then we present limitations per each study. The first two clear limitations are the number of ReLe systems studied and the number of NLP tasks the studied systems pertain to. We studied five ReLe systems: One MF system (*Model F*) for the task of knowledge base pop-

ulation, two systems using Bi-LSTMs (*ESIM* and *CE*) and one using MLPs (*DAM*) for the task of natural language inference, and one system (*GloVe*) for the task of hypernymy prediction. This means that our results and findings may not generalize to other ReLe systems and to other tasks. Thus, we believe a wider range of ReLe systems should be studied, namely systems based on Neural Turing Machines, Memory Networks, Gated Recurrent Units, among others. Furthermore, we believe tasks such as question answering, machine translation, reading comprehension, among others may also benefit from the type of studies we carried out in this work. Moreover, we believe our work may also help to better understand ReLe systems in tasks that lie in the intersection of NLP with computer vision, such as visual question answering.

Future work that we leave open to the NLP community is to apply the all the analyses presented here to a single ReLe system of interest. We applied three different types of analysis to different ReLe systems; however, did not analyze a single system using the three different approaches. We believe that doing so would provide a valuable understanding of any target system. As we saw in the previous chapters, a representation-level analysis provides the researcher with a model of the decision process of a target system. A behavioral analysis provides an explanation of what external and internal factors influence the predictions of the target system. An internal analysis verifies whether a target system is able to encode some type of information. Thus, by having a target system analyzed under these three approaches, we would better understand why the system behaves as such, from an algorithmic and stimulus-response perspectives, and what information it has capture.

The above proposition for future work leads to another open problem, namely to integrate these three approaches into an unified explanation. Even though an isolationist approach is usually taken across scientific disciplines,[4] we believe efforts should be put on this endeavour. The results obtained would certainly aid to our understanding of ReLe systems; for example, if we are to unify analyses from cognitive and behavioral sciences, we would probably obtain a model that explains

[4]We do not find very often in the literature works that present unified results from more than two disciplines due to the great complexity in doing so.

the decision process of the target system while taking into account the internal and external factors that influence the decisions of the system; in other words, we would obtain a model that explains the role that such factors play in the decision process of the target system. Take another scenario were we unify analyses from cognitive science and neuroscience; in this case, we would probably obtain explanations of how the decision process of the system takes place in the internal mechanisms of the target system (similar to work done in the cognitive neuroscience discipline.)

Another proposition for future work is to use more types of analysis to study ReLe systems. Two other widely-used analyses in the cognitive science and neuroscience disciplines are *functional analysis* and *mechanistic analysis* (see Section 3.1.2 for a description.) A functional analysis explains how a system's capacity works in terms of sub-capacities –functions– and the way they are organized; for example, long-term memory in humans may be analyzed in terms of sub-capacities, namely memory encoding, storage, and retrieval. In a similar way, we can study ReLe systems by pinpointing possible capacities and explain them in terms of sub-capacities, providing a functional explanation of how ReLe systems work. Complementary to a functional analysis is a mechanistic analysis; in this type of analysis, we aim to understand how the internal components of the system are organized and how they lead to the observed behavior.

The NLP community may also benefit from following the above propositions to study specific capabilities of ReLe systems that today are hypothetical. For example, while behavioral and internal analyses may aid to figure out if a system encodes *reasoning*, a functional analysis may help to diagram how a system performs such a capacity. Also, behavioral analyses, similar to those presented in this work in Chapter 5 and in Section 3.2.1, may help to figure out if a system *understands* natural language, and if so, a functional analysis, again, may help to provide an explanation in the form of a functional diagram of how such capacity works. This endeavour may help to either prove or *demystify* the capabilities of ReLe systems.

Another limitation of this work is that we only provide qualitative analyses of ReLe systems and we do not provide ways of improving the systems based on our

findings. This is clearly a line of future work: How can we debug a system using its proxy model? How can we improve the robustness of a system by understanding which factors influence its behavior? How can we better use the word embeddings of a system that we know has captured certain linguistic phenomena? We believe the findings of this thesis will allow us, in future work, to address these questions. Furthermore, based on our analysis from Chapter 5 where we find a bias in the data, and base on our analysis from Chapter 6 where we show how to analyze a dataset, how to test its usefulness, and how to build better datasets, we propose as future work to use our analyses to better study, evaluate and construct datasets in different fields of NLP such as natural language inference, question answering, machine translation, among others; i.e., we propose that each new dataset created is a) tested for biases and b) verified to align to the phenomena required by the task; and we finally propose that new versions of datasets created are evaluated following our work.

We now propose future work based on each of our case studies.

7.4.1 First Case Study from Chapter 4

We envisage two major lines of future research based on our work in this chapter. First, we propose to test the proxy models on users. We believe that is important to test how easy is to users to understand the explanations from a proxy model; after all, using a proxy model to understand the predictions of a black-box system is the final objective of an interpretability analysis. Second, we propose to use a proxy models to debug a black-box system. A proxy model shows an approximation of how a black-box system makes a decision. For example, our Bayesian network is a model of the decision process of *Model F*; the BN explains a prediction via a sequence of probabilistic entailments that goes from the observed input variables up to the response of *Model F*; then, by observing the BN, we are able to spot spurious entailments, such as Daniel Kahneman reviewing movies due to the fact that we won the Nobel price. Then, we are able to apply a mechanism to delete such a spurious correlation; for example, by re-training *Model F* with the constrain of not allowing the above correlation. Overall, we believe that proxy models can be helpful for

debugging black-box systems and it is pertinent to evaluate their usefulness with users.

7.4.2 Second Case Study from Chapter 5

Derived from the results and findings in this case study, we envisage two proposals for future research. First, we propose to improve upon the methods we used, which we borrowed from behavioral science. More concretely, we propose to study more methods to better control for confounding factors; as we mentioned in Chapter 5, after we applied our transformation to input data some confounding factors appeared in the new test samples, such as new interactions of words. Thus, we propose to investigate both more methods for controlling factors and more statistical tests to assess the impact of such factors on the response of the system under study. Moreover, we add to our proposal of future work the problem of analyzing how confounding factors interact with each other. In our case study, we analyzed each of the confounding factors independently of each other; however, factors may influence each other. Second, we propose to investigate new ways for transforming data in order to obtain new challenging test sets. Our transformation was very simple and it generated a type of test set; however, it is important to apply new transformations in order to generate new types of data that can help us to study the ReLe systems under different angles. For example, by swapping more than one word pair in the instances, or by adding a new word in specific places within instances, we can obtain new test sets where new confounding factors and new phenomena to test (such as meronymy) may arise and thus we can further learn about the behavior of ReLe systems.

7.4.3 Third Case Study from Chapter 6

Two short-term future works in this case study are to analyze more hypernymy datasets and more word embeddings models. Furthermore, we also propose to extract other semantic relations, such as antonymy and meronymy, from word embeddings and to analyze their respective datasets using our framework. By doing so, we would be more certain as to what extent word embeddings are able to capture

semantic information. Moreover, we propose to pay more attention to works from neuroscience in order to enrich our studies of ReLe systems by applying both invasive and non-invasive methods that can help us to understand how ReLe systems encode other types of information, how their memory is organized, and how certain behavioral patterns arise from the internal mechanisms.

Bibliography

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8. URL http://dl.acm.org/citation.cfm?id=645920.672836.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Record*, 1993.

Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2 edition, 2010. ISBN 026201243X, 9780262012430.

Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373 – 389, 1995. ISSN 0950-7051. doi: https://doi.org/10.1016/0950-7051(96)81920-4. URL http://www.sciencedirect.com/science/article/pii/0950705196819204. Knowledge-based neural networks.

Homa B. Hashemi and Rebecca Hwa. An evaluation of parser robustness for ungrammatical sentences. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1765–1774, Austin, Texas, November 2016. Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1182.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja

Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1859912.

Bart Baesens, Rudy Setiono, Christophe Mues, and Jan Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Manage. Sci.*, 49(3):312–329, March 2003. ISSN 0025-1909. doi: 10.1287/mnsc.49.3.312.12739. URL http://dx.doi.org/10.1287/mnsc.49.3.312.12739.

Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics, 2011.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France, April 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E12-1004.

David Barrett. Functional analysis and mechanistic explanation. *Synthese*, 191(12): 2695–2714, Aug 2014. ISSN 1573-0964. doi: 10.1007/s11229-014-0410-9. URL https://doi.org/10.1007/s11229-014-0410-9.

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2613–2621. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6339-measuring-neural-net-robustness-with-constraints.pdf.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.

Emily M. Bender, Dan Flickinger, Stephan Oepen, and Yi Zhang. Parser evaluation over local and non-local deep dependencies in a large corpus. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 397–408, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL `http://dl.acm.org/citation.cfm?id=2145432.2145479`.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.50.

Lucy P Birkett and Nicholas E Newton-Fisher. How abnormal is the behaviour of captive, zoo-living chimpanzees? *PloS one*, 6(6):e20101, 2011.

Ned Block. The computer model of the mind. In Daniel N. Osherson and Edward E. Smith, editors, *Thinking (Vol. 3): An Invitation to Cognitive Science*, pages 247–289. MIT Press, Cambridge, MA, USA, 1990. ISBN 0-262-65035-5. URL `http://dl.acm.org/citation.cfm?id=103067.103007`.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4349–4357. Curran Associates, Inc., 2016.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/D15-1075`.

Bruce G Buchanan and Edward A Feigenbaum. Dendral and meta-dendral: Their applications dimension. *Artificial intelligence*, 11(1):5–24, 1978.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788613. URL http://doi.acm.org/10.1145/2783258.2788613.

Alexander M. Chan, Eric Halgren, Ksenija Marinkovic, and Sydney S. Cash. Decoding word and category-specific spatiotemporal representations from meg and eeg. *NeuroImage*, 54(4):3028 – 3039, 2011. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2010.10.073. URL http://www.sciencedirect.com/science/article/pii/S1053811910013819.

Olivier Chapelle and Ya Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 1–10, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526711. URL http://doi.acm.org/10.1145/1526709.1526711.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1082.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen.

Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1152.

David Maxwell Chickering. *Learning Bayesian Networks is NP-Complete*, pages 121–130. Springer New York, New York, NY, 1996. ISBN 978-1-4612-2404-4. doi: 10.1007/978-1-4612-2404-4_12. URL https://doi.org/10.1007/978-1-4612-2404-4_12.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512. Curran Associates, Inc., 2016.

Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 24–30, Cambridge, MA, USA, 1995. MIT Press. URL http://dl.acm.org/citation.cfm?id=2998828.2998832.

Robert Cummins. Functional analysis. *The Journal of Philosophy*, 72(20):741–765, 1975.

Ido Dagan and Oren Glickman. Probabilistic textual entailment: generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France, 2004.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii, 2009. doi: 10.1017/S1351324909990209.

A.S d'Avila Garcez, K Broda, and D.M Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intel-*

ligence, 125(1):155 – 207, 2001. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(00)00077-1. URL http://www.sciencedirect.com/science/article/pii/S0004370200000771.

Marian Stamp Dawkins. Behaviour as a tool in the assessment of animal welfare. *Zoology*, 106(4):383 – 387, 2003. ISSN 0944-2006. doi: https://doi.org/10.1078/0944-2006-00122. URL http://www.sciencedirect.com/science/article/pii/S0944200604701130.

Pedro Domingos. Knowledge discovery via multiple models. *Intelligent Data Analysis*, 2(1):187 – 202, 1998. ISSN 1088-467X. doi: https://doi.org/10.1016/S1088-467X(98)00023-7. URL http://www.sciencedirect.com/science/article/pii/S1088467X98000237.

Marek J. Druzdzel. Explanation in probabilistic systems: Is it feasible? will it work. *Proceedings of the Fifth International Workshop on Intelligent Information Systems (WIS-96)*, pages 12 – 24, 1996.

W. Frank Epling and W. David Pierce. The basic importance of applied behavior analysis. *The Behavior Analyst*, 9(1):89–99, Apr 1986. ISSN 2196-8918. doi: 10.1007/BF03391932. URL https://doi.org/10.1007/BF03391932.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, Aug 2017. ISSN 1573-0565. doi: 10.1007/s10994-017-5663-3. URL https://doi.org/10.1007/s10994-017-5663-3.

Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press., Cambridge, MA, 1998.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *ACM Transactions on Information Systems*, pages 116–131. Association for Computing Machinery, January 2002.

J. R. Firth. A synopsis of linguistic theory, 1930-1955. *A Synopsis of Linguistic Theory 1930-1955*, pages 1–32, 1957. URL www.scopus.com. Cited By :77.

Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, March 2014. ISSN 1931-0145. doi: 10.1145/2594473.2594475. URL http://doi.acm.org/10.1145/2594473.2594475.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, Nov 1997. ISSN 1573-0565. doi: 10.1023/A:1007465528199. URL https://doi.org/10.1023/A:1007465528199.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1113.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. International World Wide Web Conferences Steering Committee, 2013.

A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 148–155, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers

Inc. ISBN 1-55860-555-X. URL http://dl.acm.org/citation.cfm?id=2074094.2074112.

D. C. Geary. *The origin of mind: Evolution of brain, cognition, and general intelligence*. American Psychological Association, Washington DC, US, 2005.

Victor Gijsbers. Explanatory pluralism and the (dis) unity of science: the argument from incompatible counterfactual consequences. *Frontiers in psychiatry*, 7:32, 2016.

Baggio Giosuè, Lambalgen Michiel, and Hagoort Peter. Logic as marr's computational level: Four case studies. *Topics in Cognitive Science*, 7(2):287–298, 2014. doi: 10.1111/tops.12125. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12125.

Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, pages 345–420, 2016.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J. Goodfellow, Jonathon Shlens, and hristian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR2015)*. CBLS, 2015.

Thomas L Griffiths, Charles Kemp, and Joshua B Tenenbaum. Bayesian models of cognition. In Ron Sun, editor, *The Cambridge handbook of computational cognitive modeling*. Cambridge University Press, 2008.

Stephen R Grimm. The goal of explanation. *Studies in History and Philosophy of Science Part A*, 41(4):337–344, 2010.

Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. Distributional vectors encode referential attributes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 12–21. Association

for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1002. URL http://www.aclweb.org/anthology/D15-1002.

Stephen B. Hager. The diversity of behavior. *Nature Education Knowledge*, 4(66), 2010.

Robert R. Hampton. Rhesus monkeys know when they remember. *Proceedings of the National Academy of Sciences*, 98(9):5359–5362, 2001. ISSN 0027-8424. doi: 10.1073/pnas.071600998. URL http://www.pnas.org/content/98/9/5359.

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL http://dx.doi.org/10.1080/00437956.1954.11659520.

Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009. ISSN 1041-4347. doi: 10.1109/TKDE.2008.239. URL http://dx.doi.org/10.1109/TKDE.2008.239.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Tomoyasu Horikawa and Yukiyasu Kamitani. Hierarchical neural representation of dreamed objects revealed by brain decoding with deep neural network features. *Frontiers in Computational Neuroscience*, 11(4), 08 2017.

Jin Huang and C. X. Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 17(3):299–310, March 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.50.

Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141

– 154, 2011. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2010.12.003. URL http://www.sciencedirect.com/science/article/pii/S0167923610002368.

Pierre Isabelle, Colin Cherry, and George Foster. A challenge set approach to evaluating machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2476–2486, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

H. Jacobsson, S. L. Frank, and D. Federici. Automated abstraction of dynamic neural systems for natural language processing. In *2007 International Joint Conference on Neural Networks*, pages 1446–1451, Aug 2007. doi: 10.1109/IJCNN.2007.4371171.

Henrik Jacobsson. Rule extraction from recurrent neural networks: A taxonomy and review. *Neural Comput.*, 17(6):1223–1263, June 2005. ISSN 0899-7667. doi: 10.1162/0899766053630350. URL http://dx.doi.org/10.1162/0899766053630350.

Davy Janssens, Geert Wets, Tom Brijs, Koen Vanhoof, Theo Arentze, and Harry Timmermans. Improving performance of multiagent rule-based model for activity pattern decisions with bayesian networks. *Transportation Research Record: Journal of the Transportation Research Board*, (1894):75–83, 2004.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2011–2021, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Bickle John. Marr and reductionism. *Topics in Cognitive Science*, 7(2):299–311, 2015. doi: 10.1111/tops.12134. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12134.

David M. Kaplan. Integrating mind and brain science: A field guide. In David M.

Kaplan, editor, *Explanation and Integration in Mind and Brain Science*, chapter 1. Oxford University Press, Oxford, 2017.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. In *International Conference on Learning Representations-Workshop (ICLR2016)*. CBLS, 2016.

Hyeoncheol Kim, Tae-Sun Yoon, Yiying Zhang, Anupam Dikshit, and Su-Shing Chen. Predictability of rules in hiv-1 protease cleavage site analysis. In *Proceedings of the 6th International Conference on Computational Science - Volume Part II*, ICCS'06, pages 830–837, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34381-4, 978-3-540-34381-3. doi: 10.1007/11758525_111. URL http://dx.doi.org/10.1007/11758525_111.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389, 2010.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1378–1387, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/kumar16.html.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. Parser showdown at the wall street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D12-1096.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258, Valencia, Spain, April 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-1117.

Carmen Lacave and Francisco J. Díez. A review of explanation methods for bayesian networks. *Knowl. Eng. Rev.*, 17(2):107–127, June 2002. ISSN 0269-8889. doi: 10.1017/S026988890200019X. URL https://doi.org/10.1017/S026988890200019X.

Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Klaus-Robert Muller, and Wojciech Samek. Analyzing classifiers: Fisher vectors and deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2912–2920, 2016.

Jens Lehmann, Sebastian Bader, and Pascal Hitzler. Extracting reduced logic programs from artificial neural networks. *Applied Intelligence*, 32(3):249–266, Jun 2010. ISSN 1573-7497. doi: 10.1007/s10489-008-0142-y. URL https://doi.org/10.1007/s10489-008-0142-y.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association

for Computational Linguistics. URL https://aclweb.org/anthology/D16-1011.

Alessandro Lenci and Giulia Benotto. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 75–79, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2387636.2387650.

Benjamin Letham, Cynthia Rudin, Tyler H. McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9:1350–1371, 9 2015. ISSN 1932-6157. doi: 10.1214/15-AOAS848.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

Omer Levy, Ido Dagan, and Jacob Goldberger. Focused entailment graphs for open ie propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-1610.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1098.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N16-1082.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/980691.980696. URL https://doi.org/10.3115/980691.980696.

Charles X Ling and Robert J Yan. Decision tree with better ranking. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 480–487, 2003.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. ISSN 2307-387X. URL https://www.transacl.org/ojs/index.php/tacl/article/view/972.

Paulo J.G. Lisboa, Terence A. Etchells, Ian H. Jarman, M.S. Hane Aung, Sylvie Chabaud, Thomas Bachelot, David Perol, Thérèse Gargi, Valérie Bourdès, Stéphane Bonnevay, and Sylvie Négrier. Time-to-event analysis with artificial neural networks: An integrated analytical and rule-based study for breast cancer. *Neural Networks*, 21(2):414 – 426, 2008. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2007.12.034. URL http://www.sciencedirect.com/science/article/pii/S0893608007002638. Advances in Neural Networks Research: IJCNN '07.

H. Liu and P. Singh. Conceptnet — a practical commonsense reasoning tool-kit.

BT Technology Journal, 22(4):211–226, Oct 2004. ISSN 1573-1995. doi: 10.1023/B:BTTJ.0000047600.45421.6d. URL https://doi.org/10.1023/B:BTTJ.0000047600.45421.6d.

Sheng Liu, Shamitha Dissanayake, Sanjay Patel, Xin Dang, Todd Mlsna, Yixin Chen, and Dawn Wilkins. Learning accurate and interpretable models based on regularized random forests regression. In *BMC Systems Biology 8(Suppl 3)*, 2014.

Jorge M. Lobo, Alberto Jiménez-Valverde, and Raimundo Real. Auc: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, 17(2):145–151, 2008. ISSN 1466-8238. doi: 10.1111/j.1466-8238.2007.00358.x. URL http://dx.doi.org/10.1111/j.1466-8238.2007.00358.x.

Peter LoBue and Alexander Yates. Types of common-sense knowledge needed for recognizing textual entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 329–334, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002805.

Fernando Lopes da Silva. Eeg and meg: Relevance to neuroscience. *Neuron*, 80(5):1112 – 1128, 2013. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2013.10.017. URL http://www.sciencedirect.com/science/article/pii/S0896627313009203.

Olli J. Loukola, Clint J. Perry, Louie Coscos, and Lars Chittka. Bumblebees show cognitive flexibility by improving on an observed complex behavior. *Science*, 355 (6327):833–836, 2017. ISSN 0036-8075. doi: 10.1126/science.aag2360. URL http://science.sciencemag.org/content/355/6327/833.

Bill Maccartney. *Natural Language Inference*. PhD thesis, Stanford, CA, USA, 2009. AAI3364139.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Dragos D. Margineantu and Thomas G. Dietterich. Improved class probability estimates from decision tree models. In David D. Denison, Mark H. Hansen, Christopher C. Holmes, Bani Mallick, and Bin Yu, editors, *Nonlinear Estimation and Classification*, pages 173–188. Springer New York, New York, NY, 2003. ISBN 978-0-387-21579-2. doi: 10.1007/978-0-387-21579-2_10. URL https://doi.org/10.1007/978-0-387-21579-2_10.

D. Marr. Artificial intelligence—a personal view. *Artificial Intelligence*, 9(1):37 – 48, 1977. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(77)90013-3. URL http://www.sciencedirect.com/science/article/pii/0004370277900133.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. The MIT Press, Cambridge, MA, 2010.

David Martens, Bart Baesens, and Tony Van Gestel. Decompositional rule extraction from support vector machines by active learning. *IEEE Trans. on Knowl. and Data Eng.*, 21(2):178–191, February 2009. ISSN 1041-4347. doi: 10.1109/TKDE.2008.131. URL http://dx.doi.org/10.1109/TKDE.2008.131.

J.H. McDonald. *Handbook of Biological Statistics*. Sparky House Publishing, Baltimore, Maryland, 3 edition, 2014.

Ken McRae, George S. Cree, Mark S. Seidenberg, and Chris Mcnorgan. Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, 37(4):547–559, Nov 2005. ISSN 1554-3528. doi: 10.3758/BF03192726. URL https://doi.org/10.3758/BF03192726.

Joy Mench. Why it is important to understand animal behavior. *ILAR journal /*

National Research Council, Institute of Laboratory Animal Resources, 39:20–26, 02 1998.

Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations Workshop (ICLR201w)*. CBLS, 2013.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. Adversarial sets for regularising neural link predictors. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*, 2017. URL `http://auai.org/uai2017/proceedings/papers/306.pdf`.

Saif M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney. Computing lexical contrast. *Computational Linguistics*, 39(3):555–590, 2013.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X, 9780262018258.

W. James Murdoch and Arthur Szlam. Automatic rule extraction from long short term memory networks. In *International Conference on Learning Representations (ICLR2017)*. CBLS, 2017.

Gregory Murphy. *The big book of concepts*. The MIT Press, 2004. ISBN 0262632993.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Thomas Naselaris, Ryan J Prenger, Kendrick Kay, Michael Oliver, and Jack Gallant. Bayesian reconstruction of natural images from human brain activity. *Neuron*, 63 (6):902–915, 09 2009.

Thomas Naselaris, Kendrick N. Kay, Shinji Nishimoto, and Jack L. Gallant. Encoding and decoding in fmri. *NeuroImage*, 56(2):400 – 410, 2011. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2010.07.073. URL http://www.sciencedirect.com/science/article/pii/S1053811910010657. Multivariate Decoding and Brain Reading.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192. Association for Computational Linguistics, 2015. doi: 10.18653/v1/S15-1021. URL http://www.aclweb.org/anthology/S15-1021.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015.

Ulf H. Nielsen, Jean-Philippe Pellet, and André Elisseeff. Explanation trees for causal bayesian networks. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

Shinji Nishimoto, An T. Vu, Thomas Naselaris, Yuval Benjamini, Bin Yu, and Jack L. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641–1646, 2011.

Koichi Odajima, Yoichi Hayashi, Gong Tianxia, and Rudy Setiono. Greedy rule generation from discrete data and its use in neural network rule extraction. *Neural Networks*, 21(7):1020 – 1028, 2008. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2008.01.003. URL http://www.sciencedirect.com/science/article/pii/S0893608008000580.

T. E. Oliphant. Python for scientific computing. *Computing in Science Engineering*, 9(3):10–20, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.58.

M. Pacer, J. J. Williams, X. Chen, T. Lombrozo, and T. L. Griffiths. Evaluating computational models of explanation using human judgments. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2013.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1244.

Robert Parker and et al. English gigaword fifth edition, 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1162.

Claudia Perlich, Foster Provost, and Jeffrey S. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. *J. Mach. Learn. Res.*, 4:211–255, December 2003. ISSN 1532-4435. doi: 10.1162/153244304322972694. URL https://doi.org/10.1162/153244304322972694.

Gualtiero Piccinini and Carl Craver. Integrating psychology and neuroscience: functional analyses as mechanism sketches. *Synthese*, 183(3):283–311, Dec 2011. ISSN 1573-0964. doi: 10.1007/s11229-011-9898-4. URL https://doi.org/10.1007/s11229-011-9898-4.

JC Platt. Probabilistic outputs for support vector machines for pattern recognition. *Advances in Large margin Classifiers. Kluwer Academic Publishers, Boston*, 64: 975–1005, 1999.

Foster Provost and Pedro Domingos. Tree induction for probability-based ranking. *Mach. Learn.*, 52(3):199–215, September 2003. ISSN 0885-6125.

doi: 10.1023/A:1024099825458. URL https://doi.org/10.1023/A:1024099825458.

Fran H. Regan, Jo Hockenhull, Joy C. Pritchard, Avril E. Waterman-Pearson, and Helen R. Whay. Behavioural repertoire of working donkeys and consistency of behaviour over time, as a preliminary step towards identifying pain-related behaviours. *PLOS ONE*, 9(7):1–7, 07 2014. doi: 10.1371/journal.pone.0101877. URL https://doi.org/10.1371/journal.pone.0101877.

Marek Rei and Ted Briscoe. Looking for hyponyms in vector space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 68–77, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-1608.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939778. URL http://doi.acm.org/10.1145/2939672.2939778.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. Relation extraction with matrix factorization and universal schemas. In *Joint Human Language Technology Conference/Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, June 2013.

Laura Rimell, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 813–821, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-62-6. URL http://dl.acm.org/citation.cfm?id=1699571.1699619.

Stephen Roller and Katrin Erk. Relations such as hypernymy: Identifying and

exploiting hearst patterns in distributional vectors for lexical entailment. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2163–2172, Austin, Texas, November 2016. Association for Computational Linguistics. URL https://aclweb.org/anthology/D16-1234.

Stephen Roller, Katrin Erk, and Gemma Boleda. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1097.

Martin Roth and Robert Cummins. Two tales of functional explanation. *Philosophical Psychology*, 27(6):773–788, 2014.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2 edition, 2003. ISBN 9780137903955.

Emad W. Saad and Donald C. Wunsch. Neural network explanation using inversion. *Neural Networks*, 20(1):78 – 93, 2007. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2006.07.005. URL http://www.sciencedirect.com/science/article/pii/S0893608006001730.

Mark Sammons, V.G.Vinod Vydiswaran, and Dan Roth. "ask not what textual entailment can do for you...". In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1199–1208, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1122.

Evan Sandhaus. The new york times annotated corpus. 2008.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics,*

volume 2: Short Papers, pages 38–42, Gothenburg, Sweden, April 2014a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E14-4008.

Enrico Santus, Qin Lu, Aless, ro Lenci, and Chu-Ren Huang. Taking antonymy mask off in vector space. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, 2014b. URL http://www.aclweb.org/anthology/Y14-1018.

Sanne Schoenmakers, Markus Barth, Tom Heskes, and Marcel van Gerven. Linear reconstruction of perceived images from human brain activity. *NeuroImage*, 83:951 – 961, 2013. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2013.07.043. URL http://www.sciencedirect.com/science/article/pii/S1053811913007994.

Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *ArXiv e-prints*, 2016.

Rudy Setiono, Bart Baesens, and Christophe Mues. A note on knowledge discovery using neural networks and its application to credit card screening. *European Journal of Operational Research*, 192(1):326 – 332, 2009. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2007.09.022. URL http://www.sciencedirect.com/science/article/pii/S0377221707009290.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107057132, 9781107057135.

Sara J. Shettleworth and Jennifer E. Sutton. Do animals know what they know?

In Susan Hurley and Matthew Nudds, editors, *Rational Animals?*, chapter 11. Oxford University Press, Oxford, 2006.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Ajit P. Singh and Geoffrey J. Gordon. A unified view of matrix factorization models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ECML PKDD '08, pages 358–373, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-87480-5. doi: 10.1007/978-3-540-87481-2_24. URL http://dx.doi.org/10.1007/978-3-540-87481-2_24.

Edward E. Smith and Stephen M. Kosslyn. *Cognitive psychology: mind and brain*, volume 6. Pearson/Prentice Hall, first edition edition, 2007. ISBN 9781292022352.

Kerri Smith. Brain decoding: Reading minds. *Nature*, 502(7472):428–430, 2013.

Dilip Soman. Effects of payment mechanism on spending behavior: The role of rehearsal and immediacy of payments. *Journal of Consumer Research*, 27(4): 460–474, 2001.

Jae W Song and Kevin C Chung. Observational studies: cohort and case-control studies. *Plastic and reconstructive surgery*, 126(6):2234, 2010.

Michael Strevens. No understanding without explanation. *Studies in History and Philosophy of Science Part A*, 44(3):510 – 515, 2013. ISSN 0039-3681. doi: https://doi.org/10.1016/j.shpsa.2012.12.005. URL http://www.sciencedirect.com/science/article/pii/S003936811200115X.

Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, and Tom Mitchell. Tracking neural coding of perceptual

and semantic features of concrete nouns. *NeuroImage*, 62(1):451 – 463, 2012. ISSN 1053-8119. doi: https://doi.org/10.1016/j.neuroimage.2012.04.048. URL http://www.sciencedirect.com/science/article/pii/S1053811912004442.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR2014)*. CBLS, 2014.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1150.

Brian J Taylor and Maijorie A Darrah. Rule extraction as a formal method for the verification and validation of neural networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 2915–2920. IEEE, 2005.

Terry Therneau, Beth Atkinson, Brian Ripley, and Maintainer Brian Ripley. Package rpart, 2014.

Sebastian Thrun. Extracting rules from artificial neural networks with distributed representations. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, pages 505–512, Cambridge, MA, USA, 1994. MIT Press. URL http://dl.acm.org/citation.cfm?id=2998687.2998750.

A. B. Tickle, R. Andrews, M. Golea, and J. Diederich. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *Trans. Neur. Netw.*, 9(6):1057–1068, November 1998.

ISSN 1045-9227. doi: 10.1109/72.728352. URL http://dx.doi.org/10.1109/72.728352.

A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4577-0394-2. doi: 10.1109/CVPR.2011.5995347. URL http://dx.doi.org/10.1109/CVPR.2011.5995347.

Geoffrey G. Towell and Jude W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–101, Oct 1993. ISSN 1573-0565. doi: 10.1007/BF00993103. URL https://doi.org/10.1007/BF00993103.

Norbert Tsopze, Engelbert Mephu Nguifo, and Gilbert Tindo. Towards a generalization of decompositional approach of rule extraction from multilayer artificial neural network. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1562–1569, 07 2011.

Stephen Turner. Where explanation ends: Understanding as the place the spade turns in the social sciences. *Studies in History and Philosophy of Science Part A*, 44(3):532 – 538, 2013. ISSN 0039-3681. doi: https://doi.org/10.1016/j.shpsa.2012.12.001. URL http://www.sciencedirect.com/science/article/pii/S0039368112001112.

Peter D. Turney and Saif M. Mohammad. Experiments with three approaches to recognizing lexical entailment. *Natural Language Engineering*, 21(03):437–476, 2015.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

M.L. Vaughn. Derivation of the multilayer perceptron weight constraints for direct network interpretation and knowledge discovery. *Neural Networks*, 12(9):1259 – 1271, 1999. ISSN 0893-6080. doi: https://doi.org/

10.1016/S0893-6080(99)00062-3. URL http://www.sciencedirect.com/science/article/pii/S0893608099000623.

Wouter Verbeke, David Martens, Christophe Mues, and Bart Baesens. Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications*, 38(3):2354 – 2364, 2011. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2010.08.023. URL http://www.sciencedirect.com/science/article/pii/S0957417410008067.

Ivan Vulić, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen. Hyperlex: A large-scale evaluation of graded lexical entailment. *Computational Linguistics*, 43(4):781–835, 2017.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1158.

Robert Wall, Pádraig Cunningham, Paul Walsh, and Stephen Byrne. Explaining the output of ensembles in medical decision support on a case by case basis. *Artificial Intelligence in Medicine*, 28(2):191 – 206, 2003. ISSN 0933-3657. doi: https://doi.org/10.1016/S0933-3657(03)00056-3. URL http://www.sciencedirect.com/science/article/pii/S0933365703000563. Knowledge-based Neurocomputing in Medicine.

Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *Proceedings of Coling 2004*, pages 1015–1021, Geneva, Switzerland, Aug 23–Aug 27 2004. COLING.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the*

25th International Conference on Computational Linguistics: Technical Papers, pages 2249–2259, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL http://www.aclweb.org/anthology/C14-1212.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 271–280, Vancouver, Canada, August 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/K17-1028.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 996–1005, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL http://www.aclweb.org/anthology/I17-1100.

Jim Woodward. *Explanation*, pages 37–54. Blackwell Publishers Ltd, 2008. ISBN 9780470756614. doi: 10.1002/9780470756614.ch3. URL http://dx.doi.org/10.1002/9780470756614.ch3.

Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. doi: 10.3115/981732.981751. URL https://doi.org/10.3115/981732.981751.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR2015)*. CBLS, 2015.

Ghim-Eng Yap, Ah-Hwee Tan, and Hwee-Hwa Pang. Explaining inferences in bayesian networks. *Applied Intelligence*, 29(3):263–278, Dec 2008. ISSN 1573-

7497. doi: 10.1007/s10489-007-0093-8. URL https://doi.org/10.1007/s10489-007-0093-8.

Elahe' Yargholi and Gholam-Ali Hossein-Zadeh. Brain decoding-classification of hand written digits from fmri data employing bayesian networks. *Frontiers in Human Neuroscience*, 10, 2016. ISSN 1662-5161. doi: 10.3389/fnhum.2016.00351. URL https://www.frontiersin.org/article/10.3389/fnhum.2016.00351.

Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning*, 2015.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10590-1. doi: 10.1007/978-3-319-10590-1_53. URL https://doi.org/10.1007/978-3-319-10590-1_53.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2941–2951, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

Maayan Zhitomirsky-Geffet and Ido Dagan. Bootstrapping distributional feature vector quality. *Computational linguistics*, 35(3):435–461, 2009.

Zhi-Hua Zhou. Rule extraction: Using neural networks or for neural networks? *Journal of Computer Science and Technology*, 19(2):249–253, Mar 2004. ISSN 1860-4749. doi: 10.1007/BF02944803. URL https://doi.org/10.1007/BF02944803.

Luisa Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualising deep neural network decisions. In *International Conference on Learning Representations (ICLR2017)*. CBLS, 2017.

Quan Zou, Sifa Xie, Ziyu Lin, Meihong Wu, and Ying Ju. Finding the best classification threshold in imbalanced classification. *Big Data Research*, 2016.