Deep Generative Models for Natural Language

Harshil Bharat Shah

Department of Computer Science

University College London

This dissertation is submitted for the degree of

Doctor of Philosophy

Declaration

Abstract

Generative models aim to simulate the process by which a set of data is generated. They are intuitive, interpretable, and naturally suited to learning from unlabelled data. This is particularly appealing in natural language processing, where labels are often costly to obtain and can require significant manual input from trained annotators. However, traditional generative modelling approaches can often be inflexible due to the need to maintain tractable maximum likelihood training.

On the other hand, deep learning methods are powerful, flexible, and have achieved significant success on a wide variety of natural language processing tasks. In recent years, algorithms have been developed for training generative models that incorporate neural networks to parametrise their conditional distributions. These approaches aim to take advantage of the intuitiveness and interpretability of generative models as well as the power and flexibility of deep learning.

In this work, we investigate how to leverage such algorithms in order to develop deep generative models for natural language. Firstly, we present an attention-based latent variable model, trained using unlabelled data, for learning representations of sentences. Experiments such as missing word imputation and sentence similarity matching suggest that the representations are able to learn semantic information about the sentences. We then present an RNN-based latent variable model for performing machine translation. Trained using semi-supervised learning, our approach achieves strong results even with very limited labelled data. Finally, we present a locally-contextual conditional random field for performing sequence labelling tasks. Our method consistently outperforms the linear chain conditional random field and achieves state of the art performance on two out of the four tasks evaluated.

Impact statement

This work presents a number of potential benefits both inside and outside academia. Much of this work has been published at major machine learning conferences [Shah et al., 2018, Shah and Barber, 2018] and cited in follow-up research.

This work has contributed to furthering our understanding of using generative models for natural language processing, an area which has recently seen significant growth in interest in academia. The methods presented are likely to be beneficial for future research in areas such as using unlabelled data for representation learning and semi-supervised learning (both for language and other domains), training contextual embeddings of sequences of text, and learning from data from different modalities.

Outside academia, commercial organisations are accumulating increasing amounts of natural language data, therefore the ideas presented in this work are likely to be especially useful in industry. For example, the method presented in Chapter 4 achieves state of the art results on sequence labelling tasks which occur frequently in commercial applications. In addition, labelled data is often costly to acquire, therefore the approaches in Chapters 2 and 3, which make use of unlabelled data, could be valuable.

Contents

Introduction **13**

1 Foundations **16**

1.1 Generative modelling 16

1.1.1 Latent variables 16

1.1.2 Directed vs. undirected graphs 17

1.1.3 EM algorithm 19

1.2 Deep learning 20

1.2.1 Neural networks 20

1.2.2 Word embeddings 23

1.3 Deep generative models 25

1.3.1 SGVB 26

2 Unsupervised Representation Learning **29**

2.1 Introduction 29

2.2 Model 31

2.2.1 Generative process 32

2.2.2 Inference 35

2.3 Related work 36

2.4 Experiments 37

2.4.1 Preprocessing 37

2.4.2 Model architecture 37

2.4.3 Training 38

2.4.4 Results 39

2.4.5 Observing the generation process . . . 39
2.4.6 Generated sentences . . . 40
2.4.7 Imputing missing words . . . 41
2.4.8 Finding similar sentences . . . 42
2.5 Conclusion . . . 43

3 Generative Neural Machine Translation 46

3.1 Introduction . . . 46
3.2 Model . . . 48
3.2.1 Generative process . . . 49
3.2.2 Training . . . 50
3.2.3 Generating translations . . . 51
3.2.4 Translating with missing words . . . 52
3.2.5 Multilingual translation . . . 52
3.2.6 Semi-supervised learning . . . 53
3.3 Related work . . . 53
3.4 Experiments . . . 54
3.4.1 Data . . . 55
3.4.2 Training . . . 55
3.4.3 Results . . . 56
3.5 Conclusion . . . 60

4 Sequence Labelling 63

4.1 Introduction . . . 63
4.2 Linear chain CRFs . . . 65
4.2.1 Incorporating contextual information . . . 66
4.3 The CRF$^{\otimes}$ model . . . 68
4.3.1 Combination with bi-LSTMs . . . 70
4.4 Related work . . . 71
4.5 Datasets . . . 71
4.6 Experiments . . . 73
4.6.1 Model architectures and training . . . 73

4.6.2 Results 74
4.7 Ablation study 76
4.7.1 Results 77
4.7.2 Computational efficiency 78
4.8 Conclusion 78
4.9 Appendix 79
4.9.1 Computational efficiency 79
4.9.2 A more general parametrisation 80
4.9.3 Widening the contextual window 81

Conclusion **86**

Bibliography **89**

List of Figures

1.1 The HMM graphical model. 17

1.2 The CRF graphical model. 18

1.3 A basic latent variable model. 20

1.4 A feedforward neural network. 21

1.5 A recurrent neural network. 22

1.6 The Gen-LSTM graphical model. 25

2.1 The AUTR graphical model. Figure 2.1a shows the construction of the canvas and Figure 2.1b shows the generation of the sentence conditioned on the final canvas state. 32

2.2 Visualising the sequential construction of sentences generated from the learned model. We sample $\mathbf{z}$ from its prior, i.e. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and pass the sample through $p(\mathbf{x}|\mathbf{z}; \theta)$, visualising the canvas at several points along the way. The darkness indicates the cumulative attention that has been placed on that slot so far. 40

2.3 Visualising the cumulative attention on each of the sentence's slots for the second example of Figure 2.2. The vertical axis is the RNN time step and the horizontal axis indicates the position in the sentence. The darker the shade, the more attention has been placed on that position. 41

3.1 The GNMT graphical model 48

3.2 Test set BLEU scores on translation, by sentence length, evaluated using the model parameters trained with 4M paired sentences. 58

4.1 The graphical model of the linear chain CRF. 65
4.2 The graphical model of the CRF$^{\otimes}$. 68
4.3 The graphical model of the CRF$^{\otimes}$ combined with a bidirectional LSTM. 70
4.4 Results of the ablation study on each of the test sets. We compare the results when augmenting the CRF with locally-contextual connections, nonlinear potential functions, and both. 77
4.5 The graphical model of the CRF$^{\otimes}$-Concat. 79
4.6 The graphical model of the CRF$^{\otimes}$-Wide. 81

List of Tables

2.1 Test set results on the Book Corpus dataset. 39

2.2 Sentences sampled from the prior: $\mathbf{z}$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and passed through the generative model $p(\mathbf{x}|\mathbf{z}; \theta)$ to produce a sentence $\mathbf{x}$. . . . 42

2.3 Sentences sampled from the posterior. Conditioned on a test set input sentence $\mathbf{x}$, $\mathbf{z}$ is drawn from the inference distribution, $q(\mathbf{z}|\mathbf{x}; \phi)$ and passed through the generative model $p(\mathbf{x}'|\mathbf{z}; \theta)$ to produce a reconstruction $\mathbf{x}'$. 43

2.4 Imputing missing words in test set sentences, using the procedure described in Algorithm 2. The underlined words are considered as missing. 44

2.5 Finding the 'best matching' sentence using the latent representation. 45

3.1 Test set BLEU scores on translation for models trained with varying amounts of paired sentences. 57

3.2 Test set KL divergence values ($D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x}, \mathbf{y}; \phi)||p(\mathbf{z})]$) for the models trained with 4M paired sentences, averaged across languages. 58

3.3 An example of a long sentence translation, showing the ability of GNMT to capture long range semantics. 59

3.4 Test set BLEU scores for missing word translation. We use the model parameters trained with 4M paired sentences. 59

3.5 A randomly sampled test set missing word translation from English to Spanish. The underlined words are considered as missing. 60

3.6 Test set BLEU scores for unseen pair translation. We use the VNMT and GNMT parameters trained with 4M paired sentences. For GNMT-Multi and GNMT-Multi-SSL, we train new models with 4M paired sentences, but with the respective language pairs excluded during training. 61

4.1 Statistics of each of the datasets used. We use the standard splits for all datasets. *The CoNLL 2000 dataset does not include a validation set. We therefore randomly sample 1,000 sentences from the training set to use for validation. 72

4.2 Chunking results on the test set of the CoNLL 2000 dataset. 74

4.3 Part-of-speech tagging results on the test set of the Penn Treebank dataset. 75

4.4 Named entity recognition results on the test set of the CoNLL 2003 dataset. 75

4.5 Named entity recognition results on the test set of the OntoNotes 5.0 dataset. 76

4.6 Example sentences from the named entity recognition test sets where the CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) models make the correct predictions but the CRF(BERT) and CRF(Flair) models make the incorrect predictions. 82

4.7 Example sentences from the named entity recognition test sets where the CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) models make the incorrect predictions. 83

4.8 Computational efficiency statistics of each the models trained. The Prms column shows the number of parameters, in millions. Trn shows the average time taken, in seconds, for a training iteration with a batch of 128 sentences. Inf shows the average time taken, in seconds, to infer the most likely labels for a batch of 256 sentences. All values were computed using the same GPU. 84

4.9 Test set results with a more general parametrisation of the local context. 85

4.10 Test set results with a wider local context. 85

Introduction

Generative models posit the process by which a set of data is generated. The parameters of this process are typically learnt by maximising the likelihood of the observed data under the model. Generative models are intuitive, interpretable and are naturally suited to learning from unlabelled data. This is particularly appealing in natural language processing, where labels can be costly to obtain and can require significant manual input form trained annotators. Generative models have achieved success on a variety of tasks including part-of-speech tagging [Kupiec, 1992, Lafferty et al., 2001], named entity recognition [Bikel et al., 1999] and topic modelling [Blei et al., 2003]. However, traditional generative modelling approaches can often be inflexible due to the need to maintain tractable maximum likelihood training.

On the other hand, deep learning methods (which broadly encompass a class of approaches which use neural networks to learn the underlying structure of a set of data) are powerful and flexible. They are very popular in modern natural language processing, achieving significant success on a wide variety of tasks. Earlier methods were typically based on convolutional and recurrent networks [Hochreiter and Schmidhuber, 1997, Sutskever et al., 2014, Kim, 2014] however more recently, state of the art results have been achieved using contextual embedding models parametrised with Transformer networks [Vaswani et al., 2017, Radford et al., 2018, Devlin et al., 2019].

In recent years, algorithms have been developed for training generative models that incorporate neural networks to parametrise their conditional distributions [Kingma and Welling, 2014, Rezende et al., 2014, Burda et al., 2016]. Known as deep generative modelling, these approaches aim to take advantage of the intuitiveness and interpretability of generative models as well as the power and flexibility of deep learning. This thesis investigates how to leverage such algorithms in order to develop deep generative models for natural language.

In Chapter 1, we present the foundations for deep generative modelling. We first review traditional generative modelling techniques including latent variables and the expectation maximisation algorithm. We then review deep learning approaches including feedforward and recurrent neural networks as well as non-contextual and contextual word embeddings. Finally, we look at how generative modelling and deep learning can be combined with the stochastic gradient variational Bayes algorithm. This also includes a discussion of the 'KL collapse' phenomenon, a common problem when training such models.

A cornerstone task in natural language processing is to learn representations of sequences of text. Prior deep generative modelling approaches to this task were typically based on recurrent neural networks. However they suffered from the KL collapse phenomenon whereby the approximate posterior distribution would not learn an informative representation of a sequence of text. DRAW [Gregor et al., 2015] showed that using an attention mechanism to 'paint' locally on a canvas produces images of remarkable quality. In Chapter 2, we investigate whether a similar approach could work well for modelling natural language. We introduce a deep generative model which uses an RNN with a dynamic attention mechanism to iteratively update a canvas that parametrises the probability distribution over the sentence's text. By viewing the canvas at intermediate stages, and where the RNN is placing its attention, we gain insight into how the model constructs a sentence. As well as learning a meaningful latent representation for each sentence, the model generates coherent sentences and successfully imputes missing words.

Whilst the model in Chapter 2 learns representations of individual sentences, one may consider that using data from multiple modalities allows for richer representations to be learned than from a single modality alone. In natural language processing, the same sentence expressed in different languages offers the potential to learn a representation of the sentence's meaning. In Chapter 3, we introduce a deep generative model of sentences expressed in two languages. We use a latent variable as a language agnostic representation which is encouraged to learn the semantic meaning of the sentence. We use this model to perform multilingual translation, and leverage monolingual sentences during training. We evaluate the model on machine translation where it achieves competitive BLEU scores (particularly when the amount of paired training data is limited) and is especially effective at translating long sentences. When there are missing words in the source sentence, the model is able to use its learned representation to infer what those words may be and produce good translations accordingly.

Although representations provided by latent variable models such as those described in Chapters 2 and 3 have certain appealing properties, deterministic contextual token embeddings provided by pre-trained language models have recently become more and more popular. However the typical paradigm for using these embeddings involves 'fine-tuning' the language model on the downstream task of interest. This can be computationally expensive, depending on the footprint of the language model. If hardware constraints prevent fine-tuning, it can be necessary to design the architecture for the downstream task to extract the most useful information from the embeddings. Sequence labelling tasks are usually performed using CRFs [Lafferty et al., 2001], which model the label for a given word as dependent on its embedding as well as the labels of the neighbouring words. However it is often necessary to use the neighbouring words themselves when predicting the label for a given word. Therefore in Chapter 4 we enhance the CRF by directly incorporating the neighbouring words (local context) when predicting the label for a given word and by using deep, nonlinear potential functions. Our locally-contextual nonlinear CRFs can serve as a drop-in replacement for linear chain CRFs, and they have the same computational complexity for training and inference. We find improved results on several sequence labelling tasks, and show that both the local context and nonlinear potentials consistently provide improvements compared to CRFs.

We conclude with a review of the contributions of this dissertation and potential avenues for future work. The work presented in this thesis has been published in Shah et al. [2018], Shah and Barber [2018], Shah et al. [2021].

Chapter 1

Foundations

In this chapter, we introduce the technical concepts required for the remainder of the dissertation. We review generative models, deep learning, and how the two can be combined to provide intuitive yet powerful models of data generating processes.

1.1 Generative modelling

Generative models aim to simulate the process by which a set of data is generated. They are often represented using graphs. These are intuitive diagrams which specify the statistical dependencies between random variables. In a graph, the nodes correspond to random variables and an edge linking two nodes indicates dependence between those two variables.

Probability distributions are governed by a set of global parameters which will be denoted as θ throughout this dissertation. We do not consider the parameters as random variables but as deterministic. We learn their values using maximum likelihood estimation. To avoid clutter, we do not show the parameter set θ in our graphical models.

1.1.1 Latent variables

It is often the case that high dimensional observed data points correspond to lower dimensional manifolds within the high dimensional space. This phenomenon can be represented using latent random variables. These are variables which we assume

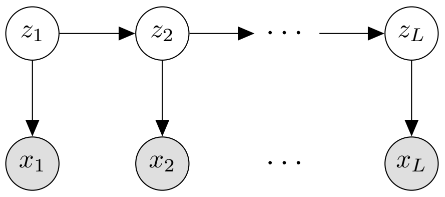


Figure 1.1: The HMM graphical model.

to exist, but are unobserved. For example in natural language processing, latent variables can be used to represent the semantic meaning of a piece of text because this is not something that is directly observed in a sequence of words but instead must be inferred. Because they are not observed, latent variables are marginalised out during maximum likelihood estimation.

Throughout, we use shaded circular nodes to denote observed variables and clear circular nodes to denote latent variables. Rhombus-shaped nodes denote deterministic functions of the input variables.

1.1.2 Directed vs. undirected graphs

In this dissertation, we consider two types of graphical model: directed and undirected. Directed graphs represent the factorisation of a distribution into conditional probabilities of variables dependent on their 'parents' in the graph. This means that they follow the chain rule of probability. In contrast, undirected graphs indicate dependence between variables but do not follow the chain rule of probability. Note that each type of graph can represent certain families of distributions that the other cannot.

1.1.2.1 Directed graphs

An example of a directed graph used in natural language processing is the Hidden Markov Model (HMM) shown in Figure 1.1 [Rabiner and Juang, 1986]. Used for unsupervised learning, the HMM posits a sequence of discrete-valued hidden states $z_1, \ldots, z_L$ for a corresponding sequence of words $x_1, \ldots, x_L$. Each hidden state z_l represents an unobserved characteristic of word x_l. The likelihood specified by this

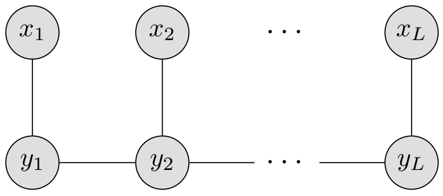


Figure 1.2: The CRF graphical model.

graph is

$$p(x_{1:L}|\theta) = \sum_{z_{1:L}} \prod_{l=1}^{L} p(z_l|z_{l-1};\theta)p(x_l|z_l;\theta). \tag{1.1}$$

In directed graphs, each node is associated with a normalised distribution. In this example, for each hidden state z_l the distribution is $p(z_l|z_{l-1};\theta)$ and for each word x_l the distribution is $p(x_l|z_l;\theta)$.

Because the words and hidden states are all discrete, the conditional distributions $p(z_l|z_{l-1};\theta)$ and $p(x_l|z_l;\theta)$ are simply parametrised using probability tables, the entries of which make up the parameter set θ.

1.1.2.2 Undirected graphs

An example of an undirected graph used in natural language processing is the linear chain conditional random field (CRF) shown in Figure 1.2 [Lafferty et al., 2001]. The CRF is used for modelling a sequence of words $x_1, \ldots, x_L$ and a corresponding sequence of linguistic tags $y_1, \ldots, y_L$ (e.g. part of speech). It assumes that each label y_l is correlated with its neighbours y_{l-1} and y_{l+1} but does not assume a left-to-right ordering like the HMM in Figure 1.1. The conditional likelihood specified by this graph is

$$p(y_{1:L}|x_{1:L};\theta) = \frac{\prod_{l=1}^{L} \psi(y_{l-1}, y_l;\theta)\eta(y_l, x_l;\theta)}{\sum_{y_{1:L}} \prod_{l=1}^{L} \psi(y_{l-1}, y_l;\theta)\eta(y_l, x_l;\theta)}. \tag{1.2}$$

The terms $\psi(y_{l-1}, y_l;\theta)$ and $\eta(y_l, x_l;\theta)$ are referred to as the potentials, which are positive valued functions. Unlike the directed setting, in undirected graphs the potentials themselves are not necessarily normalised. Rather, the distribution as a whole is normalised by summing over the product of the potentials, as in the denominator in Equation (1.2). This term is called the normalisation constant.

1.1.3 EM algorithm

Consider the graph in Figure 1.3. For simplicity, we assume that the latent variable z is continuous (however the algorithm also applies when z is discrete). The log likelihood is

$$\log p(x|\theta) = \log \int p(z|\theta)p(x|z;\theta)dz. \tag{1.3}$$

Depending on the functional forms of $p(z|\theta)$ and $p(x|z;\theta)$, this integral may not be tractable, in which case the likelihood cannot easily be computed. In this case, the expectation maximisation (EM) algorithm can be used [Dempster et al., 1977]. It introduces an 'inference' distribution, $q(z|x)$ in order to lower bound the log likelihood using Jensen's inequality as follows:

$$\begin{aligned}
\log p(x) &= \log \int p(z|\theta)p(x|z;\theta)dz && (1.4)\\
&= \log \int \frac{q(z|x)}{q(z|x)} p(z|\theta)p(x|z;\theta)dz && (1.5)\\
&\geq \int q(z|x) \log \frac{p(z|\theta)p(x|z;\theta)}{q(z|x)} dz && (1.6)\\
&\equiv \mathcal{L}(x;q,\theta). && (1.7)
\end{aligned}$$

The EM algorithm then iteratively increases this lower bound on the log likelihood by alternating between:

- **E-step**: maximise $\mathcal{L}(x;q,\theta)$ with respect to $q(z|x)$ while holding the parameters θ fixed.
- **M-step**: maximise $\mathcal{L}(x;q,\theta)$ with respect to the parameters θ while holding $q(z|x)$ fixed.

Note that $\mathcal{L}(x;q,\theta)$ can be expressed as

$$\mathcal{L}(x;q,\theta) = \int q(z|x) \log \frac{p(z|\theta)p(x|z;\theta)}{q(z|x)} dz \tag{1.8}$$

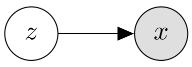


Figure 1.3: A basic latent variable model.

$$= \int q(z|x) \log \frac{p(x|\theta)p(z|x;\theta)}{q(z|x)} dz \tag{1.9}$$

$$= \int q(z|x) \log p(x|\theta) dz + \int q(z|x) \log \frac{p(z|x;\theta)}{q(z|x)} dz \tag{1.10}$$

$$= \log p(x|\theta) - D_{\mathrm{KL}}[q(z|x)||p(z|x;\theta)]. \tag{1.11}$$

$D_{\mathrm{KL}}[q||p] \geq 0$ and $D_{\mathrm{KL}}[q||p] = 0$ if and only if $q = p$. Therefore in the E-step, $\mathcal{L}(x; q, \theta)$ is maximised by setting $q(z|x) = p(z|x;\theta)$. After an E-step, the lower bound is equal to the log likelihood. This means that after an E-step and M-step, the log likelihood is guaranteed to increase.

1.2 Deep learning

Deep learning broadly encompasses a class of methods which use neural networks to learn the underlying structure of a set of data.

1.2.1 Neural networks

A neural network is a sequence of layers, each of which applies a (usually nonlinear) deterministic function to the output of the previous layer [LeCun et al., 2015, Schmidhuber, 2015, Goodfellow et al., 2016]. Each layer typically has one or more parameters, known as weights.

The simplest type of neural network is a feedforward network, as shown in Figure 1.4. A feedforward network with D layers maps input $\mathbf{x}$ to output $\mathbf{h}_D$ as follows:

$$\mathbf{h}_0 = \mathbf{x} \tag{1.12}$$

$$\mathbf{h}_1 = \sigma_1(\mathbf{W}_1 \cdot \mathbf{h}_0) \tag{1.13}$$

$$\mathbf{h}_2 = \sigma_2(\mathbf{W}_2 \cdot \mathbf{h}_1) \tag{1.14}$$

$$\dots \tag{1.15}$$

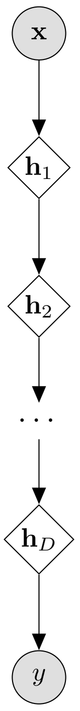


Figure 1.4: A feedforward neural network.

$$\mathbf{h}_D = \sigma_D(\mathbf{W}_D \cdot \mathbf{h}_D). \tag{1.16}$$

The values $\mathbf{h}_1, \dots, \mathbf{h}_D$ are the hidden states (or activations), the matrices $\mathbf{W}_1, \dots,$ $\mathbf{W}_D$ are the weights, and the functions $\sigma_1, \dots, \sigma_D$ are nonlinear transformations, e.g. ReLU [Glorot et al., 2011]. The output $\mathbf{h}_D$ can then be used for the task of interest. For example in classification, the softmax function can be applied to $\mathbf{h}_D$ to provide probabilities for a class label y as follows:

$$p(y = a|\mathbf{x}) \propto \exp(\mathbf{w}_a^\mathsf{T} \mathbf{h}_D). \tag{1.17}$$

The optimal values for the weights of a neural network are usually learned using stochastic gradient descent or one of its variants [Nesterov, 1983, Kingma and Ba, 2015].

Neural networks have been shown to be universal function approximators [Hornik, 1991]. This has allowed them to achieve significant success when modelling complex datasets.

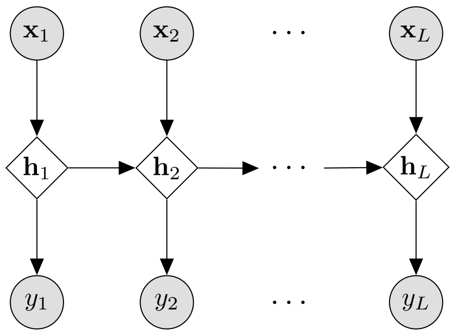


Figure 1.5: A recurrent neural network.

1.2.1.1 Recurrent neural networks

Feedforward networks are generally suitable when the input is a single observation (e.g. an image). When the input is a sequence (e.g. a sequence of words making up a sentence/document), recurrent neural networks (RNNs) are more appropriate. RNNs, as shown in Figure 1.5, process each step of the input sequentially. They maintain a hidden state which is a representation of the information from all of the past inputs.

Given a sequence of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_L$, an RNN computes the sequence of hidden states for $l = 1, \ldots, L$ as

$$\mathbf{h}_l = \sigma_l(\mathbf{W}_x \cdot \mathbf{x}_l + \mathbf{W}_h \cdot \mathbf{h}_{l-1}). \tag{1.18}$$

With RNNs, the weight matrices $\mathbf{W}_x$ and $\mathbf{W}_h$ are shared across time steps. The probabilities for each class label y_l for $l = 1, \ldots, L$ can then be computed as

$$p(y_l = a|\mathbf{x}_{1:L}) \propto \exp(\mathbf{w}_a^\intercal \mathbf{h}_l). \tag{1.19}$$

Long Short-Term Memory RNNs often struggle to model very long sequences. The long-short term memory (LSTM) [Hochreiter and Schmidhuber, 1997] was developed to address this issue by using a gating mechanism as follows:

$$\mathbf{i}_l = \sigma_i(\mathbf{W}_{xi} \cdot \mathbf{x}_l + \mathbf{W}_{hi} \cdot \mathbf{h}_{l-1}) \tag{1.20}$$

$$\mathbf{f}_l = \sigma_f(\mathbf{W}_{xf} \cdot \mathbf{x}_l + \mathbf{W}_{hf} \cdot \mathbf{h}_{l-1}) \tag{1.21}$$

$$\mathbf{o}_l = \sigma_o(\mathbf{W}_{xo} \cdot \mathbf{x}_l + \mathbf{W}_{ho} \cdot \mathbf{h}_{l-1}) \tag{1.22}$$

$$\mathbf{c}_l = \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \sigma_c(\mathbf{W}_{xc} \cdot \mathbf{x}_l + \mathbf{W}_{hc} \cdot \mathbf{h}_{l-1}) \tag{1.23}$$

$$\mathbf{h}_l = \mathbf{o}_l \odot \sigma_h(\mathbf{c}_l). \tag{1.24}$$

where $\odot$ denotes elementwise multiplication. The state $\mathbf{c}_l$ is referred to as the memory cell. The forget gate $\mathbf{f}_l$ determines how much of the memory to remove and the input gate $\mathbf{i}_l$ determines the new information to commit to the memory. The output gate $\mathbf{o}_l$ controls the dependence of the hidden state $\mathbf{h}_l$ on the memory cell.

1.2.2 Word embeddings

In natural language processing, words (or tokens) are usually considered as elements of a set known as the vocabulary. The naive way to represent a word would be to encode it as a one-hot vector. This is a vector whose length is the size of the vocabulary, with 0s everywhere except in one position (corresponding to that word's position in the vocabulary), where the value is 1.

However, one-hot word representations present several problems. The size of the vocabulary is often in the tens or hundreds of thousands, therefore encoding each word as a one-hot vector can be prohibitively expensive. Also, the model parameters for words that do not occur frequently in the training data could be poorly estimated. In addition, one-hot representations do not convey any similarity information about words. Once the model has been trained, it cannot work with words which were not present in the training data.

Instead of using one-hot vectors, word embeddings encode words using lower-dimensional, real-valued vectors. They are also known as distributed representations because the semantic information is spread throughout the dimensions of the vectors [Bengio et al., 2003, Collobert and Weston, 2008]. We typically find that words with similar meanings have representations which are close together in embedding space [Mikolov et al., 2013a,b].

Word embeddings can either be trained with the task of interest or pre-trained using an unlabelled corpus. When trained together with the task, the embedding vector for each word is simply a parameter of the model whose value is learned according to the task's objective function. However in modern natural language processing, the far more common paradigm is to use pre-trained embeddings. This has been shown to offer significant improvements over embeddings learned from scratch [Turian et al., 2010].

Pre-trained word embeddings are usually trained based on the word co-occurrence statistics from a large corpus of text. In this dissertation, we focus on GloVe embeddings [Pennington et al., 2014]. GloVe embeddings are built on the idea that the co-occurrence probabilities contain information about semantic meaning. $\mathbf{X}$ denotes the co-occurrence matrix, i.e. $\mathbf{X}_{i,j}$ is the number of times word j occurs in the context of word i. $p_{i,j} = \frac{\mathbf{X}_{i,j}}{\sum_k \mathbf{X}_{i,k}}$ is the probability that word j occurs in the context of word i. Pennington et al. [2014] use the example of $i =$ "ice" and $j =$ "steam". They suggest (and show empirically) that words k related to "ice" and not to "steam" (e.g. $k =$ "solid") should have a high ratio $\frac{p_{i,k}}{p_{j,k}}$ and vice versa. Therefore, they train the GloVe embeddings to minimise the least-squares objective

$$J = \sum_{i,j} f(\mathbf{X}_{i,j})(\mathbf{w}_i^{\mathsf{T}} \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log \mathbf{X}_{i,j})^2 \tag{1.25}$$

where $\mathbf{w}_i$ and b_i are the word vector and bias of word i, $\tilde{\mathbf{w}}_j$ and $\tilde{b}_j$ are the context vector and bias of word j, and $f(\cdot)$ is a function that assigns relatively lower weight to rare and frequent co-occurrences. After training, the final embedding for word i is given by

$$\mathbf{e}_i = [\mathbf{w}_i; b_i] + [\tilde{\mathbf{w}}_i; \tilde{b}_i] \tag{1.26}$$

where $[\mathbf{x}; \mathbf{y}]$ denotes the concatenation of $\mathbf{x}$ and $\mathbf{y}$.

1.2.2.1 Contextual embeddings

Traditional word embeddings, such as GloVe described above, are 'non-contextual'. This means that the embedding for any given word is the same regardless of the context (e.g. sentence) it appears in. Recently however, contextual embeddings have become more popular, having improved the state of the art on several natural language processing tasks. Contextual embeddings modify the representation for

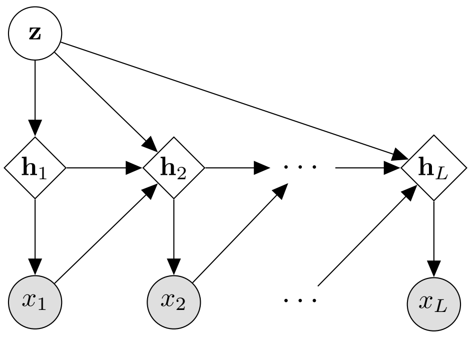


Figure 1.6: The Gen-LSTM graphical model.

each word depending on the other words in the sequence. This is particularly useful because many words can have different meanings in different contexts (e.g. “bank” can refer to the land alongside a river or a financial institution).

A popular contextual embedding model is BERT [Devlin et al., 2019]. BERT operates on sub-word tokens rather than on words. This means that the sentence is split into units which may be smaller than words. For example, the words {“waited”, “waiter”, “waiting”, “waits”} may be tokenised as {“wait + ed”, “wait + er”, “wait + ing”, “wait + s”}, all sharing the same stem. The tokenised sentences are then fed into a large Transformer network [Vaswani et al., 2017] which is trained based on two unsupervised tasks:

- **Masked language modelling:** A percentage of the input tokens are masked and the model is tasked to predict them based on the unmasked ones.
- **Next sentence prediction:** Two sentences are fed into the Transformer network and the model is tasked to predict whether or not the second follows the first in the original corpus.

Once trained, a sentence can be tokenised and fed into the network and either a sequence of word embeddings or a single sentence embedding (typically using the network’s final layer) can be returned as output.

1.3 Deep generative models

The ideas from deep learning discussed in Section 1.2 can be combined with the generative models from Section 1.1 to form powerful, flexible and intuitive models of data generating processes.

One such example is the model presented by Bowman et al. [2016], which we refer to as Gen-LSTM. Shown in Figure 1.6, Gen-LSTM is a generative model of a sentence $\mathbf{x}$ with words $x_1, \dots, x_L$. It posits a sentence-level latent variable $\mathbf{z}$ with a standard Gaussian prior

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{1.27}$$

The latent $\mathbf{z}$ and the previous word in the sentence x_{l-1} are used as the inputs to an LSTM whose states are $\mathbf{h}_1, \dots, \mathbf{h}_L$. The probability for each word conditioned on the previous words and the latent representation is

$$p(x_l = v | x_{1:l-1}, \mathbf{z}; \theta) \propto \exp(\mathbf{e}(v)^{\mathsf{T}} \mathbf{W} \cdot \mathbf{h}_l) \tag{1.28}$$

where $\mathbf{e}(v)$ is the embedding of word v and $\mathbf{W}$ is a parameter of the model. This model is trained using maximum likelihood, where the likelihood is

$$p(\mathbf{x}|\theta) = \int p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}; \theta) d\mathbf{z}. \tag{1.29}$$

This integral is intractable. In addition, the posterior distribution $p(\mathbf{z}|\mathbf{x}; \theta)$ cannot easily be computed, therefore the EM algorithm cannot directly be applied. Stochastic gradient variational Bayes is an approximate maximum likelihood algorithm which can be used in such situations.

1.3.1 SGVB

Stochastic gradient variational Bayes (SGVB) [Kingma and Welling, 2014, Rezende et al., 2014] parametrises an inference distribution $q(\mathbf{z}|\mathbf{x}; \phi)$ with parameters ϕ. It then maximises the same lower bound on the log likelihood as in the EM algorithm with respect to both the generative parameters θ and inference parameters ϕ as follows:

$$\log p(\mathbf{x}) \geq \int q(\mathbf{z}|\mathbf{x}; \phi) \log \frac{p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{x}; \phi)} \tag{1.30}$$

$$= \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x};\phi)||p(\mathbf{z})] \tag{1.31}$$

$$\equiv \mathcal{L}(\mathbf{x};\theta,\phi). \tag{1.32}$$

This bound is commonly referred to as the evidence lower bound (ELBO). The first term in Equation (1.31) is known as the 'reconstruction' term, and the second is the KL divergence from the inference distribution to the prior. The ELBO is maximised using Monte Carlo integration as follows:

$$\nabla_{\theta,\phi}\mathcal{L}(\mathbf{x};\theta,\phi) \simeq \left[\frac{1}{S}\sum_{s=1}^{S}\nabla_{\theta,\phi}p(\mathbf{x}|\mathbf{z}^{(s)};\theta)\right] - \nabla_{\phi}D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x};\phi)||p(\mathbf{z})] \tag{1.33}$$
$$\text{where } \mathbf{z}^{(s)} \sim q(\mathbf{z}|\mathbf{x};\phi).$$

However, the gradient estimate with respect to ϕ in the first term has high variance [Paisley et al., 2012]. Therefore the so-called 'reparametrisation trick' is used instead.

Reparametrisation trick Under certain conditions outlined by Kingma and Welling [2014], a random variable $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\phi)$ can be a reparametrisation of an auxiliary variable $\boldsymbol{\epsilon}$ using a differentiable transformation $g_{\phi}(\boldsymbol{\epsilon},\mathbf{x})$ as follows:

$$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \tag{1.34}$$

$$\mathbf{z} = g_{\phi}(\boldsymbol{\epsilon},\mathbf{x}). \tag{1.35}$$

A common choice for $q(\mathbf{z}|\mathbf{x};\phi)$ is a Gaussian $\mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_{\phi}(\mathbf{x})))$, where $\boldsymbol{\mu}_{\phi}(\mathbf{x})$ and $\boldsymbol{\sigma}^2_{\phi}(\mathbf{x})$ are neural networks which take $\mathbf{x}$ as input and whose weights are denoted by ϕ. In this case

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I}) \tag{1.36}$$

$$\mathbf{z} = \boldsymbol{\mu}_{\phi}(\mathbf{x}) + \boldsymbol{\sigma}_{\phi}(\mathbf{x}) \odot \boldsymbol{\epsilon}. \tag{1.37}$$

1.3.1.1 KL collapse

In models such as Gen-LSTM, the reconstruction term of the ELBO in Equation (1.31) can achieve a high value while ignoring the latent representation $\mathbf{z}$. This phenomenon is known as ‘KL collapse’. This can happen because the representational power of the LSTM combined with the previous words $x_{1:l-1}$ is often sufficient to predict the current word x_l. When this happens, the objective encourages ϕ to be set such that $q(\mathbf{z}|\mathbf{x};\phi) = p(\mathbf{z})$ and therefore $D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x};\phi)||p(\mathbf{z})] = 0$. This eliminates any potential advantages to be had from using the latent representation. To mitigate this, Bowman et al. [2016] use the following two techniques:

- **KL divergence annealing:** The KL divergence term of the ELBO in Equation (1.31) is multiplied by a constant weight which is annealed from 0 to 1 during the early iterations of training.
- **Word dropout:** During training, a random proportion of the words are replaced with an ‘unknown’ token before being passed to the next LSTM state (i.e. in the path from x_l to $\mathbf{h}_{l+1}$ in Figure 1.6).

Other types of latent variable model such as the Wasserstein Auto-Encoder [Tolstikhin et al., 2018] do not suffer from this issue. However they typically maximise an objective which is no longer a lower bound on the log-likelihood.

Chapter 2

Unsupervised Representation Learning

The work presented in this chapter was published in [Shah et al., 2018].

In the previous chapter, we presented the foundations for deep generative models. We reviewed generative modelling concepts including latent variables and the EM algorithm. We also reviewed deep learning, a paradigm which uses neural networks to learn the underlying structure of a set of data. Finally, we looked at how to combine deep learning and generative models including the SGVB training algorithm for deep latent variable models.

In this chapter, we present a deep latent variable model for learning representations of sentences. Our approach uses an attention mechanism to iteratively update a canvas that parametrises the probability distribution over the sentence's text. Experiments such as missing word imputation and sentence similarity matching suggest that the representations are able to learn semantic information about the sentences.

2.1 Introduction

Representation learning is considered to be one of the major steps towards making progress in artificial intelligence. A good representation is one whose posterior distribution captures the underlying explanatory factors behind the observed data [Bengio et al., 2013].

As discussed in Section 1.2.2, most natural language processing models make use of distributed representations of words, known as word embeddings [Bengio et al., 2003, Collobert and Weston, 2008]. Representations can also be learned for longer spans of text, such as sentences or documents. For such representations to be useful, they should ideally be able to capture long range dependencies from the text.

Language models, such as that presented in Section 1.3, are a natural way to learn representations of sentences or documents. They can be used to generate text, inspection of which can provide insight into the quality of the learned representations. Graves [2014] models sentences using a stacked RNN architecture at the character level. The distribution of a given character is dependent on the previous ones in the sentence. However, this model does not map each sentence to a single representation. This means that even though the model can be used to generate sentences that are syntactically coherent and may show local semantic consistency, it does not encourage the sentences to have long range semantic coherence.

Bowman et al. [2016] use a word level RNN, and in order to model the long range features of the text, augment it with a sentence level latent variable. Samples from the prior produce well-formed, coherent sentences, and the model is effective at imputing missing words. However, as discussed in Section 1.3.1.1, the KL divergence term of the log-likelihood lower bound reduces to 0 during training. This implies that the model ignores the latent representation and collapses to a standard RNN language model, similar to that of Graves [2014]. Word dropout is used to alleviate this problem, and this allows the model to generate sentences with more varied vocabulary and to perform better at imputing missing words than the RNN language model.

In the context of computer vision, DRAW [Gregor et al., 2015] showed that using an attention mechanism to 'paint' locally on a canvas produces images of remarkable quality. A natural question therefore is whether a similar approach could work well for natural language. To this end we introduce the **A**ttentive **U**nsupervised **T**ext (W)**r**iter (AUTR), which is a word level generative model for text [Shah et al., 2018]. AUTR uses an RNN with a dynamic attention mechanism to iteratively update a canvas that parametrises the probability distribution over the sentence's text.

Using an attention mechanism in this way can be very powerful. It allows the model to use the RNN to focus on local parts of the sentence at each time step whilst relying on the latent representation to encode global sentence features. By viewing the canvas at intermediate stages, and where the RNN is placing its attention, we gain insight into how the model constructs a sentence. Additionally, we verify that AUTR attains competitive lower bounds on the log-likelihood whilst being computationally efficient. As well as learning a meaningful latent representation for each sentence, the model generates coherent sentences and successfully imputes missing words.

2.2 Model

AUTR models the probability distribution of a sentence $\mathbf{x}$ with L words $x_1, x_2, \ldots, x_L$ as

$$p(\mathbf{x}|\theta) = \int p(\mathbf{z}) p(\mathbf{x}|\mathbf{z};\theta) d\mathbf{z} \tag{2.1}$$

where θ denotes the set of model parameters and

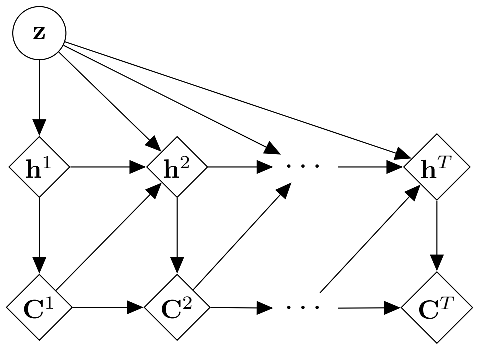
$$p(\mathbf{x}|\mathbf{z};\theta) = \prod_{l=1}^{L} p(x_l|x_{1:l-1}, \mathbf{z}; \theta). \tag{2.2}$$

For the prior of the sentence level latent variable $\mathbf{z}$, we use a standard Gaussian

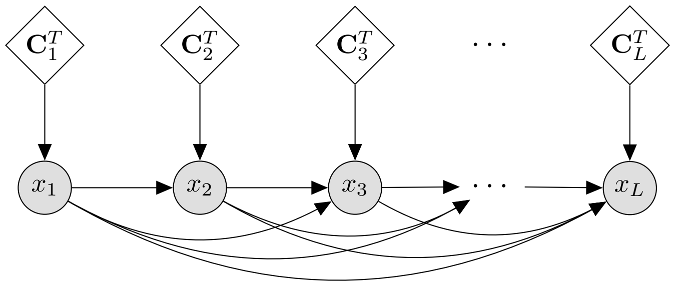
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.3}$$

To model $p(\mathbf{x}|\mathbf{z};\theta)$, AUTR uses an RNN which iteratively updates a canvas that parametrises the probability distribution over the sentence's text. Using E to denote the word embedding size, the canvas $\mathbf{C} \in \mathbb{R}^{L\times E}$ is a 2 dimensional array with L 'slots', one for each word in the sentence.

We use T to denote the number of time steps in the RNN. At each time step, an attention mechanism selects the canvas' slots to be updated; $\mathbf{C}^t$ denotes the state of the canvas at time step t. Note that T is a hyperparameter of the model.



(a) Generating the canvas. $\mathbf{z}$ is the latent representation and $\mathbf{C}^t$ denotes the state of the canvas at step t. $\mathbf{h}^t$ denotes the LSTM state at step t; it reads from the previous canvas state $\mathbf{C}^{t-1}$ and updates the canvas $\mathbf{C}^t$.



(b) Generating the sentence conditioned on the final canvas state $\mathbf{C}^T$. $\mathbf{C}_1^T, \ldots, \mathbf{C}_L^T$ are the L 'slots'.

Figure 2.1: The AUTR graphical model. Figure 2.1a shows the construction of the canvas and Figure 2.1b shows the generation of the sentence conditioned on the final canvas state.

Figure 2.1 shows the graphical model for AUTR. A summary of the generative process is given in Algorithm 1 and full details follow in Section 2.2.1.

2.2.1 Generative process

For a given sample of the latent representation $\mathbf{z}$, each RNN state $\mathbf{h}^t$ is computed as a function of this sample, as well as the previous RNN state and the canvas so far as follows:

$$\mathbf{h}^t = f(\mathbf{h}^{t-1}, [\mathbf{z}; \mathbf{C}^{t-1}]). \tag{2.4}$$

In our experiments, we use the LSTM for $f(\cdot)$ [Hochreiter and Schmidhuber, 1997]. Allowing the RNN hidden state to see what has been written to the canvas so far

Algorithm 1: AUTR generative process

```
Sample: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
Initialise: $\mathbf{h}^0 = \mathbf{0}$ and $\mathbf{C}^0 = \mathbf{0}$
for $t = 1, \dots, T$ do
    Compute: $\mathbf{h}^t = f(\mathbf{z}, \mathbf{h}^{t-1}, \mathbf{C}^{t-1})$
    Compute: $\tilde{\mathbf{g}}_l^t = \frac{\exp[(1-\sum_{\tau=1}^{t-1} \tilde{\mathbf{g}}_l^\tau)(\mathbf{W}_g \cdot \mathbf{h}^t)_l]}{\sum_{\lambda=1}^{L} \exp[(1-\sum_{\tau=1}^{t-1} \tilde{\mathbf{g}}_\lambda^\tau)(\mathbf{W}_g \cdot \mathbf{h}^t)_\lambda]}(1 - \sum_{\tau=1}^{t-1} \tilde{\mathbf{g}}_l^\tau)$
    Update: $\mathbf{C}^t = (\mathbf{1} - \tilde{\mathbf{g}}^t) \odot \mathbf{C}^{t-1} + \tilde{\mathbf{g}}^t \odot \mathbf{W}_u \cdot \mathbf{h}^t$
end
Generate: $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \mathbf{C}^T; \theta)$
```

allows the model to maintain the sentence's long range semantic coherence because the hidden state can anticipate the words that will be written at the end of the sentence and adjust the beginning accordingly, and vice versa.

Each hidden state $\mathbf{h}^t$ is then used to determine where to write (or more specifically, how 'strongly' to write to each of the canvas' slots).

2.2.1.1 Attention mechanism

We denote the 'gate' as $\mathbf{g}^t \in [0, 1]^L$ (i.e. a vector of length L with all values between 0 and 1). $\mathbf{g}^t$ determines how strongly to write to each of the L slots of the canvas. A softmax attention mechanism would be

$$\mathbf{g}_l^t = \frac{\exp[(\mathbf{W}_g \cdot \mathbf{h}^t)_l]}{\sum_{\lambda=1}^{L} \exp[(\mathbf{W}_g \cdot \mathbf{h}^t)_\lambda]} \tag{2.5}$$

where $\mathbf{W}_g$ is a parameter of the model. This would ensure that (at each time step) the attention for each slot is between 0 and 1 and the total attention across all slots is 1.

Instead, we use a modified attention mechanism as follows:

$$\tilde{\mathbf{g}}_l^t = \frac{\exp[s_l^t(\mathbf{W}_g \cdot \mathbf{h}^t)_l]}{\sum_{\lambda=1}^{L} \exp[s_\lambda^t(\mathbf{W}_g \cdot \mathbf{h}^t)_\lambda]} s_l^t \tag{2.6}$$

where

$$s_l^t = 1 - \sum_{\tau=1}^{t-1} \tilde{\mathbf{g}}_l^\tau. \tag{2.7}$$

Intuitively, in order to encourage the model to write to those slots where it hasn't yet written, we multiply each element of the softmax by one minus the accumulated attention so far (denoted by s_l^t). Then, to ensure that the cumulative attention over time applied to any of the slots is no greater than 1, we multiply the softmax itself by s_l^t. Empirically, we found this modification to perform favourably compared to the standard softmax mechanism. Note that, once a slot has been written to with a cumulative attention of 1 (i.e. $\sum_{\tau=1}^{t-1} \tilde{\mathbf{g}}_l^\tau = 1$), it cannot be updated further.

One of the key computational advantages of AUTR is that it works well with $T < L$, because it writes to multiple slots at each RNN time step. This is shown to work well empirically in Section 2.4.

2.2.1.2 Updating the canvas

The content to be written to the canvas is a linear function of the hidden state at that time step as follows:

$$\mathbf{U}^t = \mathbf{W}_u \cdot \mathbf{h}^t \tag{2.8}$$

where $\mathbf{W}_u$ is a parameter of the model. Then, using $\odot$ to denote element-wise multiplication, the canvas is updated as

$$\mathbf{C}^t = (\mathbf{1} - \tilde{\mathbf{g}}^t) \odot \mathbf{C}^{t-1} + \tilde{\mathbf{g}}^t \odot \mathbf{U}^t. \tag{2.9}$$

$g_l^t = 0$ means that the l^{th} slot from the previous time step carries over exactly to the current time step (i.e. no updating takes place), whereas $g_l^t = 1$ means that the previous values of the l^{th} slot are completely forgotten and new values are entered in their place.

We tested a fixed mechanism instead of one with attention to update the canvas, but found that the RNN didn't use the T computational steps available. At the final time step it simply overwrote everything it had previously written.

2.2.1.3 Generating the sentence

Conditioned on the final canvas, we first considered a model that generated each word independently. However, this was ineffective in our experiments since local consistency between words was lost. Therefore, we sample the sentence's text from a Markov model where the sampled word at position l depends on the l^{th} slot of the final canvas $\mathbf{C}_l^T$ and on all of the $l-1$ words that have been generated so far. We first compute the 'context' $\tilde{\mathbf{x}}_l$, which is a weighted average of the previously generated words; this is then used to modify the word probabilities that would have been assigned by the canvas alone. Specifically

$$\tilde{\mathbf{x}}_l = \sum_{l'=1}^{l-1} w_{l,l'}(\mathbf{z})\mathbf{e}(x_{l'}) \tag{2.10}$$

where $\mathbf{e}(x)$ is the embedding of word x and

$$w_{l,l'}(\mathbf{z}) = \frac{\exp[(\mathbf{W}_l \cdot \mathbf{z})_{l'}]}{\sum_{l'=1}^{l-1} \exp[(\mathbf{W}_l \cdot \mathbf{z})_{l'}]} \tag{2.11}$$

where $\mathbf{W}_l$ for $l = 1, \ldots, L$ are parameters of the model. Then, denoting $\mathbf{b}_l = \mathbf{C}_l^T + \mathbf{W}_x \cdot \tilde{\mathbf{x}}_l + \mathbf{W}_z \cdot \mathbf{z}$

$$p(x_l = a | x_{1:l-1}, \mathbf{z}; \theta) = \frac{\exp[\mathbf{e}(a)^\intercal \mathbf{b}_l]}{\sum_{v \in \mathcal{V}} \exp[\mathbf{e}(v)^\intercal \mathbf{b}_l]} \tag{2.12}$$

where $\mathbf{W}_x$ and $\mathbf{W}_z$ are parameters of the model.

2.2.2 Inference

Due to the intractability of the true posterior $p(\mathbf{z}|\mathbf{x}; \theta)$, we perform (approximate) maximum likelihood estimation using SGVB, as described in Section 1.3.1. We use a Gaussian inference distribution with parameters ϕ

$$q(\mathbf{z}|\mathbf{x}; \phi) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \text{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{x}))). \tag{2.13}$$

The mean and variance are parametrised using an LSTM, which takes as input each word embedding in order [Bowman et al., 2016]. The final LSTM state is passed through a feedforward network with two output layers to produce the mean and variance of the inference distribution.

Using this inference distribution, we train the model to maximise the ELBO

$$\mathcal{L}(\mathbf{x}; \theta, \phi) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x};\phi)}[\log p(\mathbf{x}|\mathbf{z}; \theta)] - D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x}; \phi)||p(\mathbf{z})]. \tag{2.14}$$

2.3 Related work

The AUTR canvas based model with its dynamic attention mechanism is largely inspired by DRAW [Gregor et al., 2015], which iteratively updates the canvas that parametrises the final distribution over the observed image. One of the primary differences between DRAW and AUTR is that conditioned on the final canvas, DRAW treats each pixel independently. This is not too constraining in the image domain, because small localised variations in pixel colours do not significantly affect the global quality of generated images. However in the natural language setting, because the embedding of a generated word may differ significantly from the entry in that slot of the canvas, it is necessary to condition on all previously generated words in the sentence when sampling the next word.

For natural language, deep latent variable models were popularised by Gen-LSTM, which uses an LSTM for the generative model, outputting a single word at each time step [Bowman et al., 2016]. However, this model suffers from the latent variable being ignored, as discussed in Section 1.3.1.1. It has been shown that greater reliance can be placed on the latent variable by using convolutional networks rather than RNNs in the generative model [Semeniuta et al., 2017, Yang et al., 2017]. Shen et al. [2019] develop a hierarchical model which produces more coherent and less repetitive long text compared to the shallower models.

More recently, models which make use of the Transformer architecture [Vaswani et al., 2017] have become popular for representation learning in natural language processing. Most notably, GPT [Radford et al., 2018] and BERT [Devlin et al., 2019] are large, pre-trained language models based on the Transformer architecture. In both cases, the network's internal states are used as the sentence level or word level representations for various downstream tasks, achieving significant gains over strong baselines.

2.4 Experiments

We train our model on the Book Corpus dataset [Zhu et al., 2015], which is composed of sentences from 11,038 unpublished books. We report results on language modelling tasks and compare against the LSTM model presented by Bowman et al. [2016], which we refer to as Gen-LSTM.

2.4.1 Preprocessing

We convert the text to lower case and restrict the vocabulary size to 20,000 words, keeping sentences with a maximum length of 40 words. Of the 53M sentences that meet these criteria, we use 90% for training and 10% for testing.

2.4.2 Model architecture

2.4.2.1 Generative model

For both AUTR and Gen-LSTM, we use a 50 dimensional latent representation $\mathbf{z}$ and the LSTM states have 500 units. As explained in section 2.1, one of the key advantages of AUTR is that it works well with $T < L = 40$. Therefore, we compare results with $T \in \{30, 40\}$.

2.4.2.2 Inference distribution

We train both AUTR and Gen-LSTM using SGVB. We use the architecture described in Section 2.2.2 for the inference distribution of both models; the LSTM states have 500 units.

2.4.2.3 Parameters and speed

For both models, we use 300 dimensional word embeddings, which are learned jointly with the generative and inference parameters. AUTR and Gen-LSTM have 10.9M and 10.3M parameters respectively. They take, on average, 0.19 and 0.17 seconds per training iteration and 0.06 and 0.05 seconds for sentence generation respectively.[1]

2.4.3 Training

We optimise the ELBO, shown in Equation 2.14, using stochastic gradient ascent. We train both models for 1,000,000 mini-batch iterations, using Adam with an initial learning rate of 0.0001 and mini-batches of size 200. To ensure training is fast, we use a single sample $\mathbf{z}$ per data point from the inference distribution at each iteration.

2.4.3.1 Optimisation challenges

As mentioned in Section 1.3.1.1, the KL divergence term of the ELBO can collapse to 0 when training models of this type.

KL divergence annealing We multiply the KL divergence term by a constant prefactor, which we anneal from 0 to 1 over the first 50,000 iterations of training [Bowman et al., 2016, Sønderby et al., 2016].

Word dropout To encourage Gen-LSTM to make better use of the latent representation, Bowman et al. [2016] randomly drop out a proportion of the words when training the generative RNN. Without this, they find that the model collapses to a simple RNN language model which ignores the latent representation. Following this, when training Gen-LSTM, we randomly replace 30% of the words with the unknown token. In order to assess whether or not AUTR needs the dropout mechanism to prevent the KL divergence term from collapsing to 0, we train it both with 30% dropout and without any dropout.

[1]These values are for AUTR with $T = 40$.

Model		ELBO	KL	PPL
Gen-LSTM (30% dropout)		-52.3	7.1	41.9
AUTR (no dropout)	$T = 30$	-50.7	8.0	37.4
	$T = 40$	-50.7	7.8	37.4
AUTR (30% dropout)	$T = 30$	-51.6	14.0	39.9
	$T = 40$	-51.5	13.8	39.6

Table 2.1: Test set results on the Book Corpus dataset. We report the ELBO, the contribution to the ELBO from the KL divergence term ($D_{\text{KL}}[q(\mathbf{z}|\mathbf{x};\phi)||p(\mathbf{z})]$), and the perplexity (PPL). For the ELBO, higher is better, and for the perplexity, lower is better.

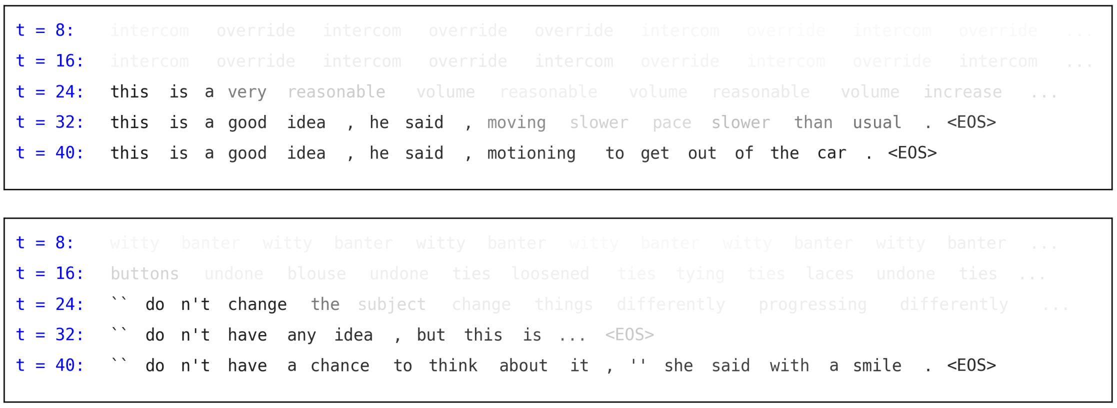
2.4.4 Results

We report test set results on the Book Corpus dataset in Table 2.1. We evaluate the ELBO on the test set by drawing 1,000 samples of the latent vector $\mathbf{z}$ per data point. We see that AUTR, both with $T = 30$ and $T = 40$, trained with or without dropout, achieves a higher ELBO and lower perplexity than Gen-LSTM. Importantly, AUTR (trained with and without dropout) relies more heavily on the latent representation than Gen-LSTM, as is shown by the larger contribution to the ELBO from the KL divergence term. As explained in Section 1.3.1.1, if a model isn't taking advantage of the latent vector $\mathbf{z}$, the loss function drives it to set $q(\mathbf{z}|\mathbf{x};\phi)$ equal to the prior $p(\mathbf{z})$ (disregarding $\mathbf{x}$), yielding a KL divergence of zero.

2.4.5 Observing the generation process

Conditioned on a sampled $\mathbf{z}$, we would like to know the most likely sentence, i.e. $\arg\max_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{z};\theta)$. However, because each word depends on all of the words generated before it, this optimisation has a computationally intractable memory requirement. We therefore perform this maximisation approximately using beam search [Reddy, 1977].

In Figure 2.2, we show examples of how the canvas changes as the RNN makes updates to it. At each time step, we take the state of the canvas and plot the sequence of words found using beam search with a beam size of 15. In Figure 2.3, we plot an example of the cumulative attention that has been placed on each of the



```
t = 8:   intercom  override  intercom  override  override  intercom  override  intercom  override  ...
t = 16:  intercom  override  intercom  override  intercom  override  intercom  override  intercom  ...
t = 24:  this  is  a  very  reasonable  volume  reasonable  volume  reasonable  volume  increase  ...
t = 32:  this  is  a  good  idea  ,  he  said  ,  moving  slower  pace  slower  than  usual  .  <EOS>
t = 40:  this  is  a  good  idea  ,  he  said  ,  motioning  to  get  out  of  the  car  .  <EOS>
```

```
t = 8:   witty  banter  witty  banter  witty  banter  witty  banter  witty  banter  witty  banter  ...
t = 16:  buttons  undone  blouse  undone  ties  loosened  ties  tying  ties  laces  undone  ties  ...
t = 24:  ``  do  n't  change  the  subject  change  things  differently  progressing  differently  ...
t = 32:  ``  do  n't  have  any  idea  ,  but  this  is  ...  <EOS>
t = 40:  ``  do  n't  have  a  chance  to  think  about  it  ,  ''  she  said  with  a  smile  .  <EOS>
```

Figure 2.2: Visualising the sequential construction of sentences generated from the learned model. We sample $\mathbf{z}$ from its prior, i.e. $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and pass the sample through $p(\mathbf{x}|\mathbf{z};\theta)$, visualising the canvas at several points along the way. The darkness indicates the cumulative attention that has been placed on that slot so far.

canvas' slots at each RNN time step.

In Figure 2.3, the model's attention appears to spend the first 15 time steps to decide how long the sentence will be (19 words in this case), and then spends the remaining 25 time steps filling in the words from left to right (even though it is not restricted to do so). It is notable that the model is able to dynamically adjust the length of the sentences by moving the end-of-sentence token and either inserting or deleting words as it sees fit. The model is also able to notice sentence features such as the open quotation marks at $t = 32$ in the second example of Figure 2.2, which it subsequently closes at $t = 40$.

2.4.6 Generated sentences

In Tables 2.2a and 2.2b, we show samples of text generated from the prior distributions for AUTR and Gen-LSTM, and in Table 2.3, we show posterior reconstructions using the inference distributions. Once again, the sentences shown are found using beam search with a beam size of 15. In both models, sampling from the prior often produces syntactically coherent sentences. However, the AUTR samples appear to show better long range semantic consistency. This is likely due to the canvas feedback mechanism when computing the RNN hidden states, and the ability of the model to place attention on the entire sentence at each time step. When attempting to

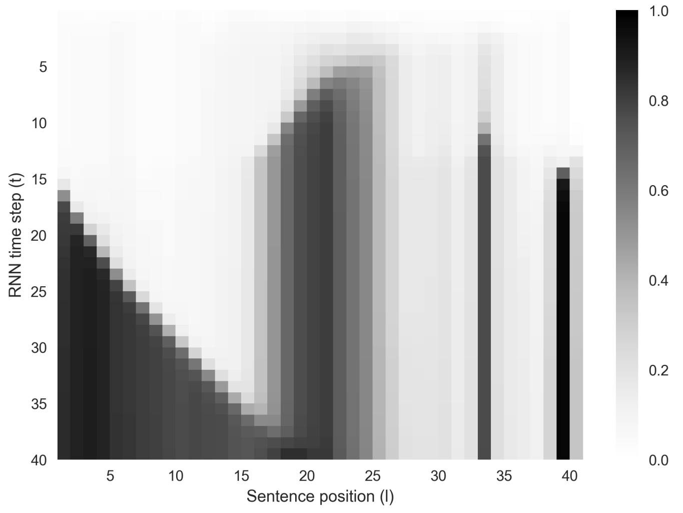


Figure 2.3: Visualising the cumulative attention on each of the sentence's slots for the second example of Figure 2.2. The vertical axis is the RNN time step and the horizontal axis indicates the position in the sentence. The darker the shade, the more attention has been placed on that position.

```
Algorithm 2: Imputing missing words
Make initial 'random' guess for missing words;
while not converged do
    E-like step: Sample $\{\mathbf{z}^{(s)}\}_{s=1}^{S}$ from $q(\mathbf{z}|\mathbf{x};\phi)$ where $\mathbf{x}$ is latest guess for
                 sentence;
    M-like step: Choose missing words in $\mathbf{x}$ to maximise
                 $\frac{1}{S}\sum_{s=1}^{S}\log p(\mathbf{x}_{\text{vis}},\mathbf{x}_{\text{miss}}|\mathbf{z}^{(s)};\theta)$ using beam search;
end
```

reconstruct a sentence, it appears that both models' latent representations capture information about meaning and length. For example, in the second sentence of Table 2.3, both models are able to recognise that there is a question. Qualitatively, the AUTR latent representation appears to learn meaning better than Gen-LSTM, as evident in several of the examples in Table 2.3.

2.4.7 Imputing missing words

AUTR's latent sentence representation makes it particularly effective at imputing missing words. To impute missing words, we use an iterative procedure inspired by the EM algorithm [Dempster et al., 1977]. We increase a variational lower bound on the log-likelihood of the visible and missing data, i.e. $\log p(\mathbf{x}_{\text{vis}},\mathbf{x}_{\text{miss}}|\theta)$, by iterating

AUTR
“do you have any idea how much i love you when i’m with you?”
“if he didn’t want to kill me,” he said, but he was trying to keep distance.
hundreds of thousands of stars rose above, reflecting the sky above the horizon.
“that sounds like a good idea,” he said, as his voice trailed off.

(a) Sentences sampled from the trained AUTR model.

Gen-LSTM
but i didn’t want to think of any other way to get it.
when i reach the top of the stairs, i feel of sight of my back into the doors open the door swings.
i had no idea what i was going to do, but i was wrong.
“you’re going to look at least one of course, but i have been in the most of course,” he said.

(b) Sentences sampled from the trained Gen-LSTM model.

Table 2.2: Sentences sampled from the prior: $\mathbf{z}$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and passed through the generative model $p(\mathbf{x}|\mathbf{z};\theta)$ to produce a sentence $\mathbf{x}$.

between an E-like and M-like step, as described in Algorithm 2. The M-like step treats the missing words as model parameters, and hence (approximately, using beam search) maximises the lower bound with respect to them.

We drop 30% of the words from each of the test set sentences and run Algorithm 2 with 50 different initialisations for the missing words. We select the resulting imputation with the highest bound on the log-likelihood. AUTR successfully imputes 34.1% of the missing words, whilst Gen-LSTM achieves 31.9%. Sampled missing word imputations for AUTR and Gen-LSTM are shown in Table 2.4.

2.4.8 Finding similar sentences

To compare the quality of the latent representations under each model, we take a sentence $\tilde{\mathbf{x}}$ from the test set and compute the mean of its posterior distribution, $\boldsymbol{\mu}_\phi(\tilde{\mathbf{x}})$. We then find the ‘best matching’ sentence $\mathbf{x}^*$ in the remainder of the test set, which satisfies

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\neq\tilde{\mathbf{x}}} \log p(\mathbf{x}|\mathbf{z} = \boldsymbol{\mu}_\phi(\tilde{\mathbf{x}});\theta). \tag{2.15}$$

If the latent representation does indeed capture the sentence’s meaning, $\mathbf{x}^*$ ought to

Input ($\mathbf{x}$)	Reconstruction (AUTR)	Reconstruction (Gen-LSTM)
unable to stop herself, she briefly, gently, touched his hand.	unable to stop herself, she leaned forward, and touched his eyes.	unable to help her, and her back and her into my way.
why didn't you tell me?	why didn't you tell me?	why didn't you tell me?"
a strange glow of sunlight shines down from above, paper white and blinding, with no heat.	the light of the sun was shining through the window, illuminating the room.	a tiny light on the door, and a few inches from behind him out of the door.
he handed her the slip of paper.	he handed her a piece of paper.	he took a sip of his drink.

Table 2.3: Sentences sampled from the posterior. Conditioned on a test set input sentence $\mathbf{x}$, $\mathbf{z}$ is drawn from the inference distribution, $q(\mathbf{z}|\mathbf{x};\phi)$ and passed through the generative model $p(\mathbf{x}'|\mathbf{z};\theta)$ to produce a reconstruction $\mathbf{x}'$.

appear qualitatively similar to $\tilde{\mathbf{x}}$.

We show some examples of the best matching sentences using AUTR and Gen-LSTM in Table 2.5; we see that the AUTR latent representations are generally successful at capturing the sentences' meanings and are able to learn sentence features such as tense and gender as well.

2.5 Conclusion

We have introduced the **A**ttentive **U**nsupervised **T**ext (W)**r**iter (AUTR), a latent variable model which uses a canvas that is dynamically updated by an attention mechanism to write natural language sentences. We visualise this canvas at intermediate stages to understand the sentence generation process. We find that the model achieves a higher ELBO on the Book Corpus dataset, and relies more heavily on the latent representation compared to a purely RNN-based generative model. In addition, we verify that it is able to generate coherent sentences as well as impute missing words effectively.

We have shown that the idea of using a canvas-based mechanism to generate text is very promising and presents plenty of avenues for future research. These

Input ($\mathbf{x}$)	Imputation (AUTR)	Imputation (Gen-LSTM)
"i want to draw you again," he says.	"i want to see you again," he said.	"i want to see you again," he said.
i believe the lie, and so i survive another day.	i believe the lie, and so will survive another day.	do believe the lie too and so will survive another day.
he was inside a house made of cheese.	he was inside a house made of cheese.	it is inside a house made a cheese.
i could have saved more of them if we had realized back then.	i would have saved more of them if we had realized back there.	i should have saved more of them than we had realized back then.

Table 2.4: Imputing missing words in test set sentences, using the procedure described in Algorithm 2. The underlined words are considered as missing.

could include investigating alternatives using fixed size canvases, as well as making the number of RNN time steps dependent on the sentence's latent representation. It may also be worthwhile to consider using convolutional layers when updating the canvas. Another particularly interesting avenue of interest is to use a similar attention mechanism to parametrise the inference distribution. Finally, similarly to recent work on Transformer-based language models [Radford et al., 2018, Devlin et al., 2019], the AUTR representations could be evaluated on downstream tasks such as natural language inference and sentiment analysis.

Input ($\tilde{\mathbf{x}}$)	Best match ($\mathbf{x}^*$) (AUTR)	Best match ($\mathbf{x}^*$) (Gen-LSTM)
he wasn't ready to face the prospect of losing her when he'd only just gotten her back.	he was never going to see her again, and that was the way it had to be.	she didn't want to make any promises, no matter how much she wanted to be with him again.
i can't help but glare at her.	i can't help but smile at her.	i couldn't help but smile at him.
so i stood in the doorway of the chapel, watching it happen.	as i sat on the bench outside the hospital, i looked up.	when he reached the bottom of the hill, he slowed his pace.
dina lets a breath out on the other side of the line.	there is a long pause on the other end of the line.	he reached into his pocket and pulled out a piece of paper.

Table 2.5: Finding the 'best matching' sentence using the latent representation.

Chapter 3

Generative Neural Machine Translation

The work presented in this chapter was published in [Shah and Barber, 2018].

In the previous chapter, we discussed the benefits of using latent variable models for learning representations of sentences. We presented an approach which used an attention mechanism to iteratively update a canvas that parametrised the probability distribution over the sentence's text. Experiments such as missing word imputation and sentence similarity matching suggested that the representations were able to learn semantic information about the sentences.

In this chapter, we demonstrate the use of latent variable language models for machine translation. Specifically, we use a latent variable as a language-agnostic representation which is encouraged to learn the semantic meaning of the sentence. We train the model using both labelled and unlabelled data and perform multilingual translation. This framework significantly reduces overfitting when there is limited paired data available, and is effective for translating between pairs of languages not seen during training.

3.1 Introduction

Data from multiple modalities (e.g. an image and a caption) can be used to learn representations with more understanding about their environment compared to when only a single modality (an image or a caption alone) is available. Such representations can then be included as components in larger models which may be responsible for several tasks. However, it can often be expensive to acquire multi-modal data even when large amounts of unlabelled data may be available.

In the machine translation context, the same sentence expressed in different languages offers the potential to learn a representation of the sentence's meaning. Variational Neural Machine Translation (VNMT) [Zhang et al., 2016] attempts to achieve this by augmenting a baseline model with a latent variable intended to represent the underlying semantics of the source sentence, achieving higher BLEU scores than the baseline. However, because the latent representation is dependent on the source sentence it can be argued that it doesn't serve a different purpose to the encoder hidden states of the baseline model. As we demonstrate empirically, this model tends to ignore the latent variable therefore it is not clear that it learns a useful representation of the sentence's meaning.

We introduce Generative Neural Machine Translation (GNMT) [Shah and Barber, 2018], whose latent variable is more explicitly designed to learn the sentence's semantic meaning. Unlike the majority of neural machine translation models (which model the conditional distribution of the target sentence given the source sentence), GNMT models the joint distribution of the target sentence and the source sentence. To do this, it uses the latent variable as a language agnostic representation of the sentence, which generates text in both the source and target languages. By giving the latent representation responsibility for generating the same sentence in multiple languages, it is encouraged to learn the semantic meaning of the sentence. We show that GNMT achieves competitive BLEU scores on translation tasks, relies heavily on the latent variable and is particularly effective at translating long sentences. When there are missing words in the source sentence, GNMT is able to use its learned representation to infer what those words may be and produce good translations accordingly.

We then extend GNMT to facilitate multilingual translation whilst sharing parameters across languages. This is achieved by adding two categorical variables to the model in order to indicate the source and target languages respectively. We

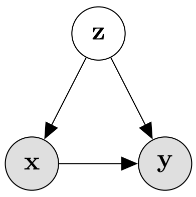


Figure 3.1: The GNMT graphical model

show that this parameter sharing helps to reduce the impact of overfitting when the amount of available paired data is limited, and proves to be effective for translating between pairs of languages which were not seen during training. We also show that by setting the source and target languages to the same value, monolingual data can be leveraged to further reduce the impact of overfitting in the limited paired data context, and to provide significant improvements for translating between previously unseen language pairs.

3.2 Model

GNMT models the joint probability of the source sentence $\mathbf{x}$ with words $x_1, \ldots, x_L$ and the target sentence $\mathbf{y}$ with words $y_1, \ldots, y_M$ by using a latent variable $\mathbf{z}$ as a language agnostic representation of the sentence. The graphical model is shown in Figure 3.1; the joint distribution is factorised as

$$p(\mathbf{x}, \mathbf{y}|\theta) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z}; \theta)p(\mathbf{y}|\mathbf{z}, \mathbf{x}; \theta)d\mathbf{z} \tag{3.1}$$

where θ denotes the set of model parameters and the prior distribution is Gaussian

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{3.2}$$

This architecture means that $\mathbf{z}$ models the commonality between the source and target sentences, which is the semantic meaning.

3.2.1 Generative process

3.2.1.1 Source sentence

To compute $p(\mathbf{x}|\mathbf{z};\theta)$, we use a model similar to that presented by Bowman et al. [2016]. The conditional probabilities for $l = 1, \ldots, L$ are

$$p(x_l = v_x | x_{1:l-1}, \mathbf{z}; \theta) \propto \exp[\mathbf{e}(v_x)^{\mathsf{T}} \mathbf{W}^x \cdot \mathbf{h}^x_l] \tag{3.3}$$

where $\mathbf{e}(v)$ denotes the embedding of word v, $\mathbf{W}^x$ is a parameter of the model and $\mathbf{h}^x_l$ is computed as

$$\mathbf{h}^x_l = f^x(\mathbf{h}^x_{l-1}, [\mathbf{z}; \mathbf{e}(x_{l-1})]). \tag{3.4}$$

In our experiments, we use the LSTM for $f^x(\cdot)$ [Hochreiter and Schmidhuber, 1997].

3.2.1.2 Target sentence

To compute $p(\mathbf{y}|\mathbf{z}, \mathbf{x};\theta)$, we modify RNNSearch [Bahdanau et al., 2015] to accommodate the latent variable $\mathbf{z}$.

Firstly, RNNSearch encodes the source sentence using a bidirectional LSTM. The encoder states for $l = 1, \ldots, L$ are computed as

$$\overrightarrow{\mathbf{h}^{enc}_l} = \overrightarrow{f^{enc}}(\overrightarrow{\mathbf{h}^{enc}_{l-1}}, \mathbf{e}(x_l)) \tag{3.5}$$

$$\overleftarrow{\mathbf{h}^{enc}_l} = \overleftarrow{f^{enc}}(\overleftarrow{\mathbf{h}^{enc}_{l+1}}, \mathbf{e}(x_l)) \tag{3.6}$$

$$\mathbf{h}^{enc}_l = [\overrightarrow{\mathbf{h}^{enc}_l}; \overleftarrow{\mathbf{h}^{enc}_l}] \tag{3.7}$$

where LSTMs are used for $\overrightarrow{f^{enc}}(\cdot)$ and $\overleftarrow{f^{enc}}(\cdot)$. Then, the decoder states for $m = 1, \ldots, M$ are computed as

$$\mathbf{h}^{dec}_m = f^{dec}(\mathbf{h}^{dec}_{m-1}, [\mathbf{e}(y_{m-1}); \mathbf{c}_m]) \tag{3.8}$$

where again, an LSTM is used for for $f^{dec}(\cdot)$ and $\mathbf{c}_m$ is a weighted sum of the encoder states and is computed as

$$\mathbf{c}_m = \sum_{l=1}^{L} \alpha_{l,m} \mathbf{h}_l^{enc} \tag{3.9}$$

$$\alpha_{l,m} = \frac{\exp(\mathbf{W}^{\alpha} \cdot [\mathbf{h}_{m-1}^{dec}; \mathbf{h}_l^{enc}])}{\sum_{\lambda=1}^{L} \exp(\mathbf{W}^{\alpha} \cdot [\mathbf{h}_{m-1}^{dec}; \mathbf{h}_{\lambda}^{enc}])} \tag{3.10}$$

where $\mathbf{W}^{\alpha}$ is a parameter of the model.

To modify RNNSearch to accommodate the latent variable $\mathbf{z}$, we simply concatenate $\mathbf{z}$ to the embedding $\mathbf{e}(x_l)$ in Equations (3.5) and (3.6) and to the embedding $\mathbf{e}(y_{m-1})$ in Equation (3.8).

Finally, the conditional probabilities for $m = 1, \ldots, M$ are

$$p(y_m = v_y | y_{1:m-1}, \mathbf{x}, \mathbf{z}; \theta) \propto \exp[\mathbf{e}(v_y)^{\mathsf{T}} \mathbf{W}^{y} \cdot \mathbf{h}_m^{dec}]. \tag{3.11}$$

3.2.2 Training

To learn the model parameters θ, we perform (approximate) maximum likelihood estimation using SGVB, as described in Section 1.3.1. To do this, we parametrise a Gaussian inference distribution with parameters ϕ

$$q(\mathbf{z}|\mathbf{x}, \mathbf{y}; \phi) = \mathcal{N}(\boldsymbol{\mu}_{\phi}(\mathbf{x}, \mathbf{y}), \mathrm{diag}(\boldsymbol{\sigma}_{\phi}^2(\mathbf{x}, \mathbf{y}))). \tag{3.12}$$

This allows us to maximise the lower bound on the log likelihood

$$\log p(\mathbf{x}, \mathbf{y}|\theta) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y};\phi)}[\log p(\mathbf{x}, \mathbf{y}|\mathbf{z}; \theta)] - D_{\mathrm{KL}}[q(\mathbf{z}|\mathbf{x}, \mathbf{y}; \phi) || p(\mathbf{z})]. \tag{3.13}$$

For the functions $\boldsymbol{\mu}_{\phi}(\mathbf{x}, \mathbf{y})$ and $\boldsymbol{\sigma}_{\phi}^2(\mathbf{x}, \mathbf{y})$, as per VNMT we first encode the source and target sentences, for $l = 1, \ldots, L$ and $m = 1, \ldots, M$, as

$$\overrightarrow{\mathbf{h}_l^{q,x}} = \overrightarrow{f^{q,x}}(\overrightarrow{\mathbf{h}_{l-1}^{q,x}}, \mathbf{e}(x_l)) \tag{3.14}$$

$$\overleftarrow{\mathbf{h}_l^{q,x}} = \overleftarrow{f^{q,x}}(\overleftarrow{\mathbf{h}_{l+1}^{q,x}}, \mathbf{e}(x_l)) \tag{3.15}$$

$$\mathbf{h}_l^{q,x} = [\overrightarrow{\mathbf{h}_l^{q,x}}; \overleftarrow{\mathbf{h}_l^{q,x}}] \tag{3.16}$$

Algorithm 3: Generating translations

```
Make initial 'random' guess for target sentence $\mathbf{y}$;
while not converged do
    E-like step: Sample $\{\mathbf{z}^{(s)}\}_{s=1}^{S}$ from $q(\mathbf{z}|\mathbf{x}, \mathbf{y}; \phi)$ where $\mathbf{y}$ is latest guess
                 for target sentence;
    M-like step: Choose words in $\mathbf{y}$ to maximise $\frac{1}{S}\sum_{s=1}^{S} \log p(\mathbf{y}|\mathbf{z}^{(s)}, \mathbf{x}; \theta)$
                 using beam search;
end
```

$$\overrightarrow{\mathbf{h}_m^{q,y}} = \overrightarrow{f^{q,y}}(\overrightarrow{\mathbf{h}_{m-1}^{q,y}}, \mathbf{e}(y_m)) \tag{3.17}$$

$$\overleftarrow{\mathbf{h}_m^{q,y}} = \overleftarrow{f^{q,y}}(\overleftarrow{\mathbf{h}_{m+1}^{q,y}}, \mathbf{e}(x_m)) \tag{3.18}$$

$$\mathbf{h}_m^{q,y} = [\overrightarrow{\mathbf{h}_m^{q,y}}; \overleftarrow{\mathbf{h}_m^{q,y}}] \tag{3.19}$$

where we use LSTMs for the functions $\overrightarrow{f^{q,x}}(\cdot)$, $\overleftarrow{f^{q,x}}(\cdot)$, $\overrightarrow{f^{q,y}}(\cdot)$ and $\overleftarrow{f^{q,y}}(\cdot)$. We then concatenate the averages of the two sets of hidden states, and use this vector to compute the mean and variance of the Gaussian inference distribution

$$\mathbf{h}^q = \left[\frac{1}{L}\sum_{l=1}^{L} \mathbf{h}_l^{q,x}; \frac{1}{M}\sum_{m=1}^{M} \mathbf{h}_m^{q,y}\right] \tag{3.20}$$

$$\boldsymbol{\mu}_\phi(\mathbf{x}, \mathbf{y}) = \mathbf{W}^\mu \cdot \mathbf{h}^q \tag{3.21}$$

$$\boldsymbol{\sigma}_\phi^2(\mathbf{x}, \mathbf{y}) = \exp(\mathbf{W}^\sigma \cdot \mathbf{h}^q). \tag{3.22}$$

3.2.3 Generating translations

Once the model has been trained, we fix the sets of parameters θ and ϕ. Then, given a source sentence $\mathbf{x}$, we want to find the target sentence $\mathbf{y}$ which maximises $p(\mathbf{y}|\mathbf{x};\theta) = \int p(\mathbf{y}|\mathbf{z}, \mathbf{x}; \theta) p(\mathbf{z}|\mathbf{x}; \theta) d\mathbf{z}$. However, this integral is intractable and so $p(\mathbf{y}|\mathbf{x};\theta)$ cannot be easily computed. Instead, because $\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x};\theta) = \arg\max_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}|\theta)$, we can perform approximate maximisation using a procedure inspired by the EM algorithm, similar to that described in Section 2.4.7. We increase a variational lower bound on $\log p(\mathbf{x}, \mathbf{y}|\theta)$ by iterating between an E-like step and an M-like step, as described in Algorithm 3.

Algorithm 4: Translating with missing words

```
Make initial 'random' guess for missing words in source sentence $\mathbf{x}_{\text{miss}}$ and
  target sentence $\mathbf{y}$;
while not converged do
    E-like step: Sample $\{\mathbf{z}^{(s)}\}_{s=1}^{S}$ from $q(\mathbf{z}|\mathbf{x},\mathbf{y};\phi)$ where $\mathbf{x}$ is latest guess
                 for source sentence and $\mathbf{y}$ is latest guess for target sentence;
    M-like step (1): Choose words in $\mathbf{x}_{\text{miss}}$ to maximise
                 $\frac{1}{S}\sum_{s=1}^{S}\log p(\mathbf{x}_{\text{vis}},\mathbf{x}_{\text{miss}}|\mathbf{z}^{(s)};\theta)$ using beam search;
    M-like step (2): Choose words in $\mathbf{y}$ to maximise
                 $\frac{1}{S}\sum_{s=1}^{S}\log p(\mathbf{y}|\mathbf{z}^{(s)},\mathbf{x};\theta)$ using beam search, where $\mathbf{x}$
                 is latest guess for source sentence;
end
```

3.2.4 Translating with missing words

Unlike architectures which model the conditional probability of the target sentence given the source sentence, $p(\mathbf{y}|\mathbf{x};\theta)$, GNMT is naturally suited to performing translation when there are missing words in the source sentence because it can use the latent representation to infer what those missing words may be.

Given a source sentence with visible words $\mathbf{x}_{\text{vis}}$ and missing words $\mathbf{x}_{\text{miss}}$, we want to find the settings of $\mathbf{x}_{\text{miss}}$ and $\mathbf{y}$ which maximise $p(\mathbf{x}_{\text{miss}},\mathbf{y}|\mathbf{x}_{\text{vis}};\theta)$. However, as in Section 3.2.3 this quantity is intractable. Therefore, we use a procedure similar to Algorithm 3. Specifically, we increase a lower bound on $\log p(\mathbf{x}_{\text{vis}},\mathbf{x}_{\text{miss}},\mathbf{y}|\theta)$, as described in Algorithm 4.

3.2.5 Multilingual translation

We extend GNMT to facilitate multilingual translation, referring to this version of the model as GNMT-Multi. We add two categorical variables to GNMT, $\mathbf{i}_x$ and $\mathbf{i}_y$ (encoded as one-hot vectors), which indicate what the source and target languages are respectively. The joint distribution becomes

$$p(\mathbf{x},\mathbf{y}|\mathbf{i}_x,\mathbf{i}_y;\theta) = \int p(\mathbf{z})p(\mathbf{x}|\mathbf{z},\mathbf{i}_x;\theta)p(\mathbf{y}|\mathbf{z},\mathbf{x},\mathbf{i}_x,\mathbf{i}_y;\theta)d\mathbf{z}. \tag{3.23}$$

This structure allows for parameters to be shared regardless of the input and output languages, and when the amount of available paired translation data is limited, this parameter sharing can significantly mitigate the risk of overfitting. The forms of the neural networks in GNMT-Multi are identical to those in GNMT (as described in Sections 3.2.1 and 3.2.2), except that $\mathbf{i}_x$ and $\mathbf{i}_y$ are now concatenated to the embeddings $\mathbf{e}(x_l)$ and $\mathbf{e}(y_m)$ respectively.

3.2.6 Semi-supervised learning

Monolingual data can be used within the GNMT-Multi framework to perform semi-supervised learning. This is simply done by setting the source and target language variables $\mathbf{i}_x$ and $\mathbf{i}_y$ to the same value, in which case the model must attempt to reconstruct the input sentence, rather than translate it. We refer to this version of the model as GNMT-Multi-SSL.

In Section 3.4, we show that when the amount of available paired translation data is limited, using monolingual data in this way further reduces overfitting compared to cross-language parameter sharing alone. Note that we are not concerned about the encoder simply copying the sentence across to the decoder, because the cross-language parameter sharing prevents this.

3.3 Related work

Whilst there have been many attempts at designing generative models of text [Bowman et al., 2016, Dieng et al., 2017, Yang et al., 2017], their usage for translation has been limited. Most closely related to our work is Variational Neural Machine Translation (VNMT) [Zhang et al., 2016], which introduces a latent variable $\mathbf{z}$ with the aim of capturing the source sentence's semantics. It models the conditional probability of the target sentence given the source sentence as $p(\mathbf{y}|\mathbf{x};\theta) = \int p(\mathbf{y}|\mathbf{z},\mathbf{x};\theta)p(\mathbf{z}|\mathbf{x};\theta)d\mathbf{z}$. The authors find that VNMT achieves improvements over modeling $p(\mathbf{y}|\mathbf{x};\theta)$ directly (i.e. without a latent variable). The primary difference compared to our work is that VNMT does not model the probability distribution of the source sentence. We believe that learning the joint distribution is a more difficult task than learning the conditional, however this is not without benefit because when learning the joint distribution, the latent variable is more explicitly encouraged to learn the semantic meaning, as shown in the examples in section 3.4. In addition, because the VNMT latent representation is dependent on the source sentence, it is not clear that it serves a different purpose to the encoder hidden states.

Also related is the work of Shu et al. [2017], which presents an approach for using unlabeled data for conditional density estimation. The authors propose a hybrid framework that regularizes the conditionally trained parameters towards the jointly trained parameters. Experiments on image modeling tasks show improvements over conditional training alone.

In work similar to GNMT-Multi, Johnson et al. [2017] perform multilingual translation whilst sharing parameters by prepending, to the source sentence, a string indicating the target language. Unlike GNMT-Multi, this approach does not indicate the source language.

There have also been various attempts to leverage monolingual data to improve translation models. Zhang and Zong [2016] use source language monolingual data and Sennrich et al. [2016] use target language monolingual data to create a synthetic dataset with which to augment the original paired dataset. This is done by passing the monolingual data through a pre-trained translation model (trained using the original paired data), thus creating a new synthetic dataset of paired data. This is combined with the original paired data to create a new, larger dataset which is used to train a new model. In both papers, the authors find that their methods obtain improvements over using paired data alone. However, these procedures do not directly integrate monolingual data into a single, unified model.

3.4 Experiments

We train GNMT, GNMT-Multi and GNMT-Multi-SSL on the 6 permutations of language pairs between English (EN), Spanish (ES) and French (FR) i.e. EN $\rightarrow$ ES, ES $\rightarrow$ EN, EN $\rightarrow$ FR, etc. We compare the performance of our models against that of VNMT, the most closely related model to our work.

The procedure for generating translations using GNMT is described in Algorithm

3. For VNMT, the conditional likelihood is

$$p(\mathbf{y}|\mathbf{x};\theta) = \int p(\mathbf{z}|\mathbf{x};\theta)p(\mathbf{y}|\mathbf{z},\mathbf{x};\theta)d\mathbf{z} \tag{3.24}$$

This can be maximized by drawing a set of samples $\{\mathbf{z}^{(s)}\}_{s=1}^{S}$ from $p(\mathbf{z}|\mathbf{x};\theta)$ and then maximizing $\frac{1}{S}\sum_{s=1}^{S} p(\mathbf{y}|\mathbf{z}^{(s)},\mathbf{x};\theta)$. This is done approximately, using beam search [Reddy, 1977].

3.4.1 Data

We use paired data provided by the Multi UN corpus [Eisele and Chen, 2010, Tiedemann, 2012]. We train each model with a small, medium and large amount of paired data, corresponding to 40K, 400K and 4M paired sentences respectively. For each language pair, we create validation sets of size 5K and test sets of size 10K paired sentences respectively. For the monolingual data used to train GNMT-Multi-SSL, we use the News Crawl articles from 2009 to 2012, provided for the WMT'13 translation task. There are 20.9M, 2.7M and 4.5M monolingual sentences for EN, ES and FR respectively.

3.4.1.1 Preprocessing

For each language, we convert all characters into lowercase and use a vocabulary of the 20,000 most common words from the paired data, replacing words outside of this vocabulary with an unknown word token. We exclude sentences which have a proportion of unknown words greater than 10% and which are longer than 50 words.

3.4.2 Training

We optimize the ELBO, shown in Equation (3.13), using stochastic gradient ascent. For all models, the latent representation $\mathbf{z}$ has 100 dimensions, each of the RNN hidden states has 1,000 units, and the word embeddings are 300-dimensional. To ensure training is fast, we use only a single sample of $\mathbf{z}$ per data point from the inference distribution at each iteration. We perform early stopping by evaluating the ELBO on the validation set every 1,000 iterations.

3.4.2.1 Optimisation challenges

As mentioned in Section 1.3.1.1, the KL divergence term of the ELBO can collapse to 0 when training models of this type.

KL divergence annealing We multiply the KL divergence term by a constant prefactor, which we anneal from 0 to 1 over the first 50,000 iterations of training [Bowman et al., 2016, Sønderby et al., 2016].

Word dropout In Equation (3.4), the dependence of the hidden state on the previous word means that the RNN can often afford to ignore the latent variable whilst still maintaining syntactic consistency between words. To prevent this from happening, during training we randomly replace the word being passed to the next RNN hidden state with the unknown word token, as suggested by Bowman et al. [2016]. This is parametrised by a drop rate, which we set to 30%. Note that this is only applied to the source sentence $\mathbf{x}$.

Word dropout significantly weakens translation performance for VNMT (since it would be applied to the target sentence $\mathbf{y}$), therefore we use KL divergence annealing when training both models, but only use word dropout when training GNMT.

3.4.3 Results

3.4.3.1 Translation

We report results on translation tasks in Table 3.1. When trained with 40K and 400K paired sentences, GNMT has a small advantage over VNMT in terms of BLEU scores across all language pairs. However, both models tend to overfit on these relatively small amounts of paired data. As a result, GNMT-Multi achieves much higher BLEU scores with both 40K and 400K paired sentences, due to the parameter sharing between languages. Adding monolingual data produces yet another significant increase in BLEU scores. In fact, GNMT-Multi-SSL trained with only 400K paired sentences achieves performance comparable with GNMT trained with 4M paired sentences. Even with 4M paired sentences, adding monolingual data is still helpful,

Paired data	System	EN → ES	ES → EN	EN → FR	FR → EN	ES → FR	FR → ES
40K	VNMT	12.45	12.30	12.20	12.98	12.19	13.44
	GNMT	13.55	12.84	12.47	13.84	13.26	14.95
	GNMT-Multi	16.32	15.36	15.99	16.92	16.80	18.21
	GNMT-Multi-SSL	23.44	22.25	20.88	20.99	22.65	24.51
400K	VNMT	33.27	31.96	27.71	27.69	28.76	31.22
	GNMT	33.87	32.75	28.55	28.98	29.41	31.33
	GNMT-Multi	40.08	38.56	35.55	37.28	36.31	38.68
	GNMT-Multi-SSL	43.96	41.63	37.37	39.66	38.09	40.79
4M	VNMT	44.10	43.03	38.06	38.56	35.28	40.27
	GNMT	44.52	43.83	37.97	38.44	35.96	40.55
	GNMT-Multi	44.43	43.91	38.02	38.67	35.57	40.79
	GNMT-Multi-SSL	45.94	45.08	39.41	40.69	38.97	42.05

Table 3.1: Test set BLEU scores on translation for models trained with varying amounts of paired sentences.

with GNMT-Multi-SSL outperforming the other models.

In Table 3.2, we report the values of the KL divergence term for the models trained with 4M paired sentences. The higher values for GNMT, GNMT-Multi and GNMT-Multi-SSL clearly indicate that these models are placing higher reliance on the latent variable than is VNMT.

BLEU by sentence length It is argued by Tu et al. [2016] that attention based translation models suffer from 'coverage' issues, particularly on long sentences, because they do not keep track of the number of times each source word is translated to a target word. However, because the latent variable in GNMT is explicitly encouraged to model the sentence's semantics, it helps to alleviate these issues. This is demonstrated in Figure 3.2 and in the example in Table 3.3, both of which show

System	VNMT	GNMT	GNMT-Multi	GNMT-Multi-SSL
D_{KL}	1.104	5.581	9.661	10.915

Table 3.2: Test set KL divergence values ($D_{\text{KL}}[q(\mathbf{z}|\mathbf{x}, \mathbf{y}; \phi)||p(\mathbf{z})]$) for the models trained with 4M paired sentences, averaged across languages.

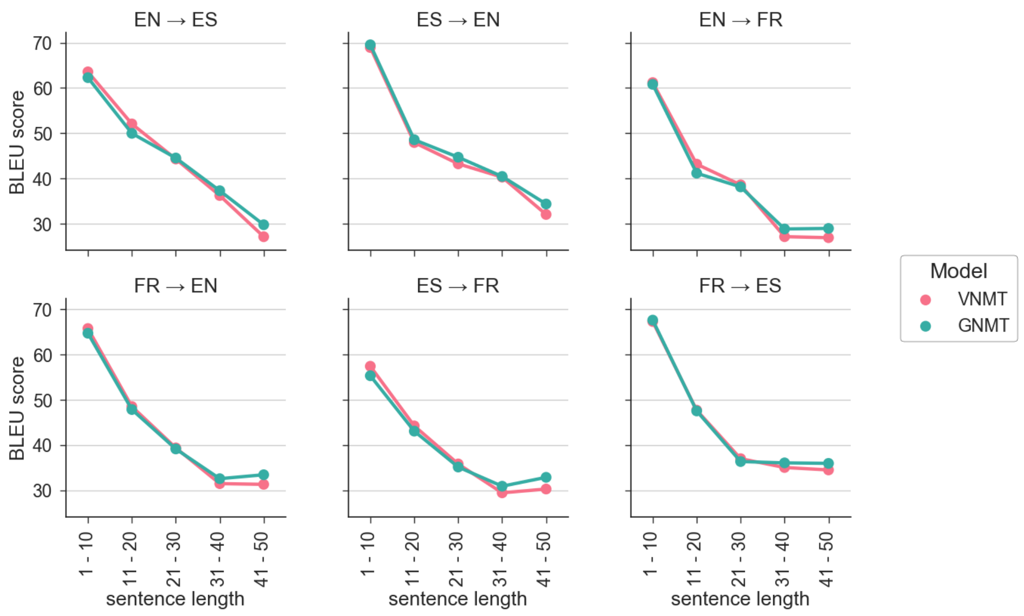


Figure 3.2: Test set BLEU scores on translation, by sentence length, evaluated using the model parameters trained with 4M paired sentences.

that GNMT tends to perform better than VNMT on long sentences, reducing the problems of translating the same phrase multiple times and neglecting to translate others at all.

3.4.3.2 Missing word translation

In order to demonstrate that GNMT does indeed learn a useful representation of the sentence's semantic meaning, we perform missing word translation (i.e. where the source sentence has missing words). The model is forced to rely on its learned representation in order to infer what the missing words could be, and then to produce a good translation accordingly.

To produce the missing word data, for each sentence we randomly sample a missing word mask where each word (independently) has a 30% chance of being missing. The procedure for generating translations using GNMT is described in Algorithm 4.

Source	dans ce décret, il met en lumière les principales réalisations de la république d'ouzbékistan dans le domaine de la protection et de la promotion des droits de l'homme et approuve le programme d'activités marquant le soixantième anniversaire de la déclaration universelle des droits de l'homme.
Target	the decree highlights major achievements by the republic of uzbekistan in the field of protection and promotion of human rights and approves the programme of activities devoted to the sixtieth anniversary of the universal declaration of human rights.
VNMT	in this regard, the decree highlights the main achievements of the uzbek republic in the promotion and promotion and protection of human rights and supports the activities of the sixtieth anniversary of the human rights of human rights.
GNMT	in this decree, it highlights the main achievements of the republic of uzbekistan on the protection and promotion of human rights and approves the activities of the sixtieth anniversary of the universal declaration of human rights.

Table 3.3: An example of a long sentence translation, showing the ability of GNMT to capture long range semantics.

System	EN → ES	ES → EN	EN → FR	FR → EN	ES → FR	FR → ES
VNMT	26.99	27.39	23.79	23.51	22.46	25.75
GNMT	33.23	33.46	29.84	28.27	29.83	33.09

Table 3.4: Test set BLEU scores for missing word translation. We use the model parameters trained with 4M paired sentences.

To generate translations using VNMT, we replace the missing words in the source sentence with the unknown word token and then conduct the same conditional likelihood maximization described in Section 3.4. The results are reported in Table 3.4. From the BLEU scores, it is evident that GNMT has a significant advantage over VNMT in this scenario, thanks to the quality of its learned representations. We show an example missing word translation in Table 3.5, where the difference in quality between GNMT and VNMT is clear.

Source	we look forward at this session to further measures to strengthen the beijing declaration and platform for action.
Target	esperamos que en este período de sesiones se adopten nuevas medidas para consolidar la declaración y la plataforma de acción de beijing.
VNMT	consideramos que el período se refiere a las medidas de fortalecimiento de la plataforma de acción de beijing.
GNMT	esperamos con interés en este período de sesiones un examen de medidas para fortalecer la declaración y la plataforma de acción de beijing.

Table 3.5: A randomly sampled test set missing word translation from English to Spanish. The underlined words are considered as missing.

3.4.3.3 Unseen language pair translation

Because GNMT-Multi shares parameters across languages, it should be naturally suited to performing translations between pairs of languages that it never saw during training. For both VNMT and GNMT, to translate, say, from English to Spanish, we first translate from English to French then from French to Spanish (because we assume the English to Spanish parameters are not available). For GNMT-Multi and GNMT-Multi-SSL, we train new models where the respective language pairs are excluded during training. Once trained, we can directly translate from the source to the target language without having to translate to an intermediate language first.

In Table 3.6, we report results on translation between previously unseen language pairs. In this context, VNMT and GNMT perform similarly in terms of BLEU scores. However, both models are consistently outperformed by GNMT-Multi (albeit only by a small amount). Using monolingual data is very effective in this context, with GNMT-Multi-SSL outperforming all other models.

3.5 Conclusion

In this chapter, we have introduced Generative Neural Machine Translation (GNMT), a latent variable architecture which aims to model the semantic meaning of the source and target sentences. For translation tasks, GNMT performs competitively

	(EN, ES) unseen		(EN, FR) unseen		(ES, FR) unseen	
System	EN → ES	ES → EN	EN → FR	FR → EN	ES → FR	FR → ES
VNMT	35.58	33.59	31.34	31.95	32.31	35.86
GNMT	35.35	33.76	31.55	31.38	32.39	35.85
GNMT-Multi	36.72	35.05	32.81	32.62	32.94	36.77
GNMT-Multi-SSL	38.80	37.43	34.79	34.98	33.57	38.11

Table 3.6: Test set BLEU scores for unseen pair translation. We use the VNMT and GNMT parameters trained with 4M paired sentences. For GNMT-Multi and GNMT-Multi-SSL, we train new models with 4M paired sentences, but with the respective language pairs excluded during training.

with a comparable conditional model, places higher reliance on the latent variable and achieves higher BLEU scores when translating long sentences. When there are missing words in the source sentence, GNMT has superior performance.

We extend GNMT to facilitate multilingual translation without adding parameters to the model. This parameter sharing reduces the impact of overfitting when the amount of available paired data is limited, and proves to be effective for translating between pairs of languages which were not seen during training. We also show that this architecture can be used to leverage monolingual data, which further reduces the impact of overfitting in the limited paired data context, and provides significant improvements for translating between previously unseen language pairs.

Whilst we chose to factorize the joint distribution as per Equation (3.1), this was not the only option we considered. The primary alternative was to use the factorisation

$$p(\mathbf{x}, \mathbf{y}|\theta) = \int p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}; \theta) p(\mathbf{y}|\mathbf{z}; \theta) d\mathbf{z}. \tag{3.25}$$

One could argue that this is in fact more natural for learning sentence semantics, since $\mathbf{z}$ wouldn't be able to rely on knowing $\mathbf{x}$ explicitly to help generate $\mathbf{y}$ through $p(\mathbf{y}|\mathbf{z}, \mathbf{x}; \theta)$. However, experimentally we found that the model struggled to generate grammatically coherent translations which also retained the source sentence's meaning.

We have shown that the idea of using the same sentence in different languages also allows for a useful latent representation to be learned. Using these sentence representations could be very promising for use in downstream tasks where 'understanding' of the environment would be helpful, e.g. question answering, dialogue generation, etc.

Chapter 4

Sequence Labelling

The work presented in this chapter was published in [Shah et al., 2021].

In the previous chapter, we presented a generative approach to machine translation which uses a latent variable as a language-agnostic representation that is encouraged to learn the semantic meaning of the sentence. We trained the model using semi-supervised learning and performed multilingual translation. This approach performed well even with limited paired data, and was effective at translating between pairs of languages not seen during training.

In this chapter, we propose locally-contextual nonlinear CRFs for sequence labelling. This approach directly incorporates information from the neighbouring embeddings when predicting the label for a given word. This model serves as a drop-in replacement for the linear chain CRF, consistently outperforming it in our ablation study. In particular, this approach outperforms the previous state of the art on two out of the four tasks evaluated.

4.1 Introduction

In natural language processing, sequence labelling tasks involve labelling every word in a sequence of text with a linguistic tag. These tasks were traditionally performed using shallow linear (in graphical structure) models such as hidden Markov models (HMMs) [Kupiec, 1992, Bikel et al., 1999] and linear chain conditional random fields (CRFs) [Lafferty et al., 2001, McCallum and Li, 2003, Sha and Pereira, 2003]. These approaches model the dependencies between adjacent word-level labels. However when predicting the label for a given word, they do not directly incorporate information from the surrounding words in the sentence (known as ‘context’). As a result, linear chain CRFs combined with deeper models which do use such contextual information (e.g. convolutional and recurrent networks) gained popularity [Collobert et al., 2011, Graves, 2012, Huang et al., 2015, Ma and Hovy, 2016].

Recently, contextual word embeddings such as those provided by pre-trained language models have become more prominent [Peters et al., 2018, Radford et al., 2018, Akbik et al., 2018, Devlin et al., 2019]. Contextual embeddings incorporate sentence-level information, therefore they can be used directly with linear chain CRFs to achieve state of the art performance on sequence labelling tasks [Yamada et al., 2020, Li et al., 2020b].

Contextual word embeddings are typically trained using a generic language modelling objective. This means that the embeddings encode contextual information which can generally be useful for a variety of tasks. However, because these embeddings are not trained for any specific downstream task, there is no guarantee that they will encode the most useful information for that task. Certain tasks such as sentiment analysis and textual entailment typically require global, semantic information about a sentence. In contrast, for sequence labelling tasks it is often the neighbouring words in a sentence which are most informative when predicting the label for a given word. Although the contextual embedding of a given word may encode information about its neighbouring words, this is not guaranteed. It can therefore be beneficial to design the sequence labelling architecture to directly extract this information from the embeddings [Bhattacharjee et al., 2020].

We therefore propose locally-contextual nonlinear CRFs for sequence labelling. Our approach extends the linear chain CRF in two straightforward ways. Firstly, we directly incorporate information from the neighbouring embeddings when predicting the label for a given word. This means that we no longer rely on each contextual embedding to have encoded information about the neighbouring words in the sentence. Secondly, we replace the linear potential functions with deep neural networks, resulting in greater modelling flexibility. Locally-contextual nonlinear

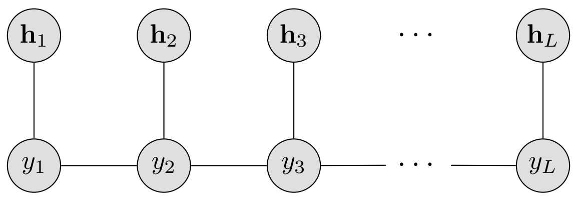


Figure 4.1: The graphical model of the linear chain CRF.

CRFs can serve as a drop-in replacement for linear chain CRFs, and they have the same computational complexity for training and inference.

We evaluate our approach on chunking, part-of-speech tagging and named entity recognition. On all tasks, our results are competitive with those of the best published methods. In particular, we outperform the previous state of the art on chunking on CoNLL 2000 and named entity recognition on OntoNotes 5.0 English. We also perform an ablation study which shows that both the local context and nonlinear potentials consistently provide improvements compared to linear chain CRFs.

4.2 Linear chain CRFs

Linear chain CRFs [Lafferty et al., 2001], as reviewed in Section 1.1.2.2, are popular for various sequence labelling tasks. These include chunking, part-of-speech tagging and named entity recognition, all of which involve labelling every word in a sequence according to a predefined set of labels.

We denote the sequence of words $\mathbf{x} = x_1, \dots, x_L$ and the corresponding sequence of labels $\mathbf{y} = y_1, \dots, y_L$. We assume that the words have been embedded using either a non-contextual or contextual embedding model; we denote the sequence of embeddings $\mathbf{h} = \mathbf{h}_1, \dots, \mathbf{h}_L$. If non-contextual embeddings are used, each embedding is a function only of the word at that time step, i.e. $\mathbf{h}_l = \mathbf{f}^{\mathrm{emb-NC}}(x_l)$. If contextual embeddings are used, each embedding is a function of the entire sentence, i.e. $\mathbf{h}_l = \mathbf{f}_l^{\mathrm{emb-C}}(\mathbf{x})$.

The linear chain CRF is shown graphically in Figure 4.1; 'linear' here refers to the graphical structure. The conditional distribution of the sequence of labels $\mathbf{y}$

given the sequence of words $\mathbf{x}$ is parametrised as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \frac{\prod_l \psi_l \eta_l}{\sum_{\mathbf{y}} \prod_l \psi_l \eta_l} \tag{4.1}$$

where θ denotes the set of model parameters. The potentials ψ_l and η_l are defined as

$$\psi_l = \psi(y_{l-1}, y_l; \theta) \tag{4.2}$$

$$\eta_l = \eta(y_l, \mathbf{h}_l; \theta). \tag{4.3}$$

The potentials are constrained to be positive and so are parametrised in log-space. Linear chain CRFs typically use log-linear potential functions

$$\log \psi_l = \mathbf{i}(y_{l-1})^{\mathsf{T}} \mathbf{A} \cdot \mathbf{i}(y_t) \tag{4.4}$$

$$\log \eta_l = \mathbf{i}(y_l)^{\mathsf{T}} \mathbf{B} \cdot \mathbf{h}_t \tag{4.5}$$

where $\mathbf{i}(y)$ denotes the one-hot encoding of y, and $\mathbf{A}$ and $\mathbf{B}$ are parameters of the model. Note that 'log-linear' here refers to linearity in the parameters. Henceforth, we refer to linear chain CRFs simply as CRFs.

4.2.1 Incorporating contextual information

When predicting the label at a given time step y_l, it is often necessary to use information from words in the sentence other than only x_l; we refer to this as contextual information. For example, below are two sentences from the CoNLL 2003 named entity recognition training set [Tjong Kim Sang and De Meulder, 2003]:

One had threatened to blow it up unless it was refuelled and they were taken to [***LOC*** London] where they intended to surrender and seek political asylum.

[***ORG*** St Helens] have now set their sights on taking the treble by winning the end-of-season premiership which begins with next Sunday's semifinal against [***ORG*** London].

In each example, the word “London” (x_l) has a different label (y_l) and so it is necessary to use contextual information to make the correct prediction.

In Figure 4.1, we see that there is no direct path from any $\mathbf{h}_j$ with $j \neq l$ to y_l. This means that in order to use contextual information, CRFs rely on either/both of the following:

- The transition potentials $\psi(y_{l-1}, y_l; \theta)$ and $\psi(y_l, y_{l+1}; \theta)$ carrying this information from the labels at positions $l-1$ and $l+1$.

- The (contextual) embedding $\mathbf{h}_l$ encoding information about words x_j with $j \neq l$.

Relying on the transition potentials may not always be effective because knowing the previous/next label is often not sufficient when labeling a given word. In the above examples, the previous/next labels indicate that they are not part of a named entity (i.e. $y_{l-1} = \text{O}$, $y_{l+1} = \text{O}$). Knowing this does not help to identify whether “London” refers to a location, an organisation, or even a person’s name.

In the first example, knowing that the previous word (x_{l-1}) is “to” and the next word (x_{l+1}) is “where” indicates that “London” is a location. In the second example, knowing that the previous word (x_{l-1}) is “against” indicates that “London” is a sports organisation. Therefore, to label these sentences correctly, a CRF relies on the embedding $\mathbf{h}_l$ to encode the identities of the neighbouring words.

However, contextual embedding models are usually trained in a manner that is agnostic to the downstream tasks they will be used for. Tasks such as sentiment analysis and textual entailment typically require global sentence-level context whereas sequence labelling tasks usually require local contextual information from the neighbouring words in the sentence, as shown in the example above. Because different downstream tasks require different types of contextual information, it is not guaranteed that task-agnostic contextual embeddings will encode the most useful contextual information for any specific task. It can therefore be beneficial to design the architecture for the downstream task to directly extract the most useful information from the embeddings.

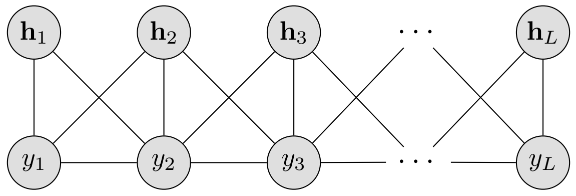


Figure 4.2: The graphical model of the CRF$^{\otimes}$.

4.3 The CRF$^{\otimes}$ model

We introduce the CRF$^{\otimes}$, which extends the CRF by directly using the neighbouring embeddings when predicting the label for a given word. We also replace the log-linear potential functions with neural networks to provide greater modelling flexibility. We call our model CRF$^{\otimes}$, where $\times$ refers to the locally-contextual structure and $\bigcirc$ refers to the nonlinear potentials.

The graphical model is shown in Figure 4.2. The conditional distribution of the labels $\mathbf{y}$ given the sentence $\mathbf{x}$ is parametrised as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \frac{\prod_l \psi_l \phi_l \eta_l \xi_l}{\sum_{\mathbf{y}} \prod_l \psi_l \phi_l \eta_l \xi_l} \tag{4.6}$$

where ψ_l and η_l are the same as in Section 4.2 and the additional potentials ϕ_l and ξ_l are defined as

$$\phi_l = \phi(y_l, \mathbf{h}_{l-1}; \theta) \tag{4.7}$$

$$\xi_l = \xi(y_l, \mathbf{h}_{l+1}; \theta). \tag{4.8}$$

With this structure, the embeddings $\mathbf{h}_{l-1}$ and $\mathbf{h}_{l+1}$, in addition to $\mathbf{h}_l$, are directly used when modelling the label y_l. This means that the model no longer relies on the embedding $\mathbf{h}_l$ to have encoded information about the neighbouring words in the sentence. We evaluate a more general parametrisation in Section 4.9.2 but find its empirical performance to be worse. We also evaluate a wider contextual window in Section 4.9.3 but find that it does not provide improved results.

Since the labels y_l are discrete, the parametrisation of $\log \psi_l$ remains the same as in Equation (4.4). However unlike the log-linear parametrisation in Equation (4.5), $\log \eta_l$ along with $\log \phi_l$ and $\log \xi_l$ are each parametrised by feedforward networks which take as input the embeddings $\mathbf{h}_{l-1}$, $\mathbf{h}_l$ and $\mathbf{h}_{l+1}$ respectively [Peng et al., 2009]

$$\log \phi_l = \mathbf{i}(y_l)^\mathsf{T} \mathbf{f}^{\phi}(\mathbf{h}_{l-1}) \tag{4.9}$$

$$\log \eta_l = \mathbf{i}(y_l)^\mathsf{T} \mathbf{f}^{\eta}(\mathbf{h}_{l}) \tag{4.10}$$

$$\log \xi_l = \mathbf{i}(y_l)^\mathsf{T} \mathbf{f}^{\xi}(\mathbf{h}_{l+1}) \tag{4.11}$$

where $\mathbf{f}^{\phi}$, $\mathbf{f}^{\eta}$ and $\mathbf{f}^{\xi}$ are the feedforward networks.

Training

We train the model to maximise $\log p(\mathbf{y}|\mathbf{x}, \theta)$ with respect to the set of parameters $\theta = \{\mathbf{A}, \phi, \eta, \xi\}$. The objective can be expressed as

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x}, \theta) = &\sum_l \left[\log \psi_l + \log \phi_l + \log \eta_l + \log \xi_l\right] \\ &- \log \sum_{\mathbf{y}} \prod_l \psi_l \phi_l \eta_l \xi_l.\end{aligned} \tag{4.12}$$

The first term is straightforward to compute. The second term can be computed using dynamic programming, analogous to computing the likelihood in HMMs [Rabiner, 1989]. We initialise

$$\alpha(y_1) = \psi_1 \eta_1. \tag{4.13}$$

Then, for $l = 2, \ldots, L$

$$\alpha(y_l) = \sum_{y_{l-1}} \alpha(y_{l-1}) \psi_l \phi_l \eta_l \xi_{l-1}. \tag{4.14}$$

Finally

$$\sum_{\mathbf{y}} \prod_l \psi_l \phi_l \eta_l \xi_l = \sum_{y_L} \alpha(y_L). \tag{4.15}$$

Denoting $\mathcal{Y}$ as the set of possible labels, the time complexity of this recursion is $O(L|\mathcal{Y}|^2)$, the same as for the CRF.

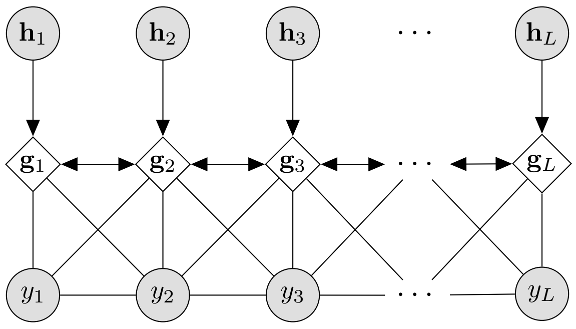


Figure 4.3: The graphical model of the CRF$^\otimes$ combined with a bidirectional LSTM.

Inference

During inference, we want to find $\mathbf{y}^*$ such that

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \ \log p(\mathbf{y}|\mathbf{x}). \tag{4.16}$$

This can be done using the Viterbi algorithm [Viterbi, 1967], which simply replaces the sum in Equation (4.14) with a max operation.

The key advantage of the CRF$^\otimes$ is that it uses enhanced local contextual information while retaining the computational tractability and parsimony of the CRF. As discussed in Section 4.2.1, sequence labelling tasks will benefit from this enhanced local context, as demonstrated empirically in the experiments and ablation study in Sections 4.6 and 4.7 respectively.

4.3.1 Combination with bi-LSTMs

Huang et al. [2015] augment the CRF with a bidirectional LSTM layer in order to use global, sentence-level contextual information when predicting the label for a given word. The CRF$^\otimes$ can similarly be combined with a bidirectional LSTM layer.

Figure 4.2 shows that with the CRF$^\otimes$, there is still no direct path from any $\mathbf{h}_{l\pm j}$ with $j \geq 2$ to y_l. Therefore, particularly when using non-contextual embeddings, encoding wider contextual information by using a bidirectional LSTM can be very useful.

The graphical model of the CRF$^\otimes$ combined with a bidirectional LSTM is shown in Figure 4.3. The sequence of embeddings $\mathbf{h}_1, \ldots, \mathbf{h}_L$ is fed to a bidirectional LSTM to produce a sequence of states $\mathbf{g}_1, \ldots, \mathbf{g}_L$. These states are then used as the inputs to the feedforward networks in Equations (4.9) to (4.11) instead of the embeddings.

4.4 Related work

There have been several approaches to directly incorporate contextual information into sequence labelling architectures. The Conv-CRF [Collobert et al., 2011], biLSTM-CRF [Huang et al., 2015], and biLSTM-CNN-CRF [Ma and Hovy, 2016] each augment the CRF with either a convolutional network, a bidirectional LSTM, or both in order to use contextual features when labelling a given word. More recently, Zhang et al. [2018] propose an alternative LSTM structure for encoding text which consists of a parallel state for each word, achieving improved results over the biLSTM-CRF for sequence labelling tasks. GCDT [Liu et al., 2019] improves an RNN-based architecture by augmenting it with a sentence-level representation which captures wider contextual information. Luo et al. [2020] learn representations encoding both sentence-level and document-level features for named entity recognition using a hierarchical bidirectional LSTM. Luoma and Pyysalo [2020] leverage the fact that BERT can represent inputs consisting of several sentences in order to use cross-sentence context when performing named entity recognition.

We compare our approach against the best of these methods in Section 4.6.

Note that in work done concurrently to ours, Hu et al. [2020] also evaluate locally-contextual parametrisations of the CRF and find that the local context consistently improves results.

4.5 Datasets

We evaluate the CRF$^{\otimes}$ on the classic sequence labelling tasks of chunking, part-of-speech tagging, and named entity recognition. We use the following datasets (summary statistics for each are shown in Table 4.1):

Chunking Chunking consists of dividing sentences into syntactically correlated parts of words according to a set of predefined chunk types. This task is evaluated

Dataset	Task	Labels	Train	Validation	Test
CoNLL 2000	Chunking	11	7,936*	1,000*	2,012
Penn Treebank	POS	45	38,219	5,527	5,462
CoNLL 2003	NER	4	14,987	3,466	3,684
OntoNotes	NER	18	59,924	8,528	8,262

Table 4.1: Statistics of each of the datasets used. We use the standard splits for all datasets.
*The CoNLL 2000 dataset does not include a validation set. We therefore randomly sample 1,000 sentences from the training set to use for validation.

using the span F1 score.

We use the CoNLL 2000 dataset [Tjong Kim Sang and Buchholz, 2000]. An example sentence from the training set is shown below:

[***NP*** He] [***VP*** reckons] [***NP*** the current account deficit] [***VP*** will narrow] [***PP*** to] [***NP*** only # 1.8 billion] [***PP*** in] [***NP*** September].

Part-of-speech tagging Part-of-speech tagging consists of labelling each word in a sentence according to its part-of-speech. This task is evaluated using accuracy.

We use the Wall Street Journal portion of the Penn Treebank dataset [Marcus et al., 1993]. An example sentence from the training set is shown below:

[***NN*** Compound] [***NNS*** yields] [***VBP*** assume] [***NN*** reinvestment] [***IN*** of] [***NNS*** dividends] [***CC*** and] [***IN*** that] [***DT*** the] [***JJ*** current] [***NN*** yield] [***VBZ*** continues] [***IN*** for] [***DT*** a] [***NN*** year].

Named entity recognition Named entity recognition consists of locating and classifying named entities in sentences. The entities are classified under a pre-defined set of entity categories.

We use the CoNLL 2003 English [Tjong Kim Sang and De Meulder, 2003] and OntoNotes 5.0 English [Pradhan et al., 2013] datasets. The CoNLL 2003 dataset consists of 4 high-level entity types while the OntoNotes 5.0 dataset involves 18 fine-grained entity types. An example from the OntoNotes 5.0 training set is shown below:

[***NORP*** Japanese] Prime Minister [***PERSON*** Junichiro Koizumi] also arrived in [***GPE*** Pusan] [***TIME*** this afternoon], beginning his [***ORG*** APEC] trip this time.

These are all important tasks in natural language processing and play vital roles in downstream tasks such as dependency parsing, question answering and relation extraction. Even small improvements on sequence labelling tasks can provide significant benefits for these downstream tasks [Nguyen and Verspoor, 2018, Park et al., 2015, Liu et al., 2017, Yamada et al., 2020].

4.6 Experiments

4.6.1 Model architectures and training

We train four different versions of the CRF$^{\otimes}$ and compare against the best published results on each dataset. The variants are as follows:

- CRF$^{\otimes}$(GloVe):
  - This uses 300-dimensional GloVe embeddings [Pennington et al., 2014] combined with the CRF$^{\otimes}$.
- CRF$^{\otimes}$(GloVe, biLSTM)
  - This uses 300-dimensional GloVe embeddings combined with a biLSTM and the CRF$^{\otimes}$. The forward and backward LSTM states each have 300 units.
- CRF$^{\otimes}$(BERT)
  - This uses the final (768-dimensional) layer of the BERT-Base model [Devlin et al., 2019] with the CRF$^{\otimes}$. BERT embeddings are defined on sub-word tokens rather than on words. Therefore if the BERT tokeniser splits a word into multiple sub-word tokens, we take the mean of the sub-word embeddings as the word embedding.

Model	F1
CRF$^{\otimes}$(GloVe)	96.12
CRF$^{\otimes}$(GloVe, biLSTM)	96.14
CRF$^{\otimes}$(BERT)	97.40
CRF$^{\otimes}$(Flair)	**97.52**
Liu et al. [2019]	97.30
Clark et al. [2018]	97.00
Akbik et al. [2018]	96.72

Table 4.2: Chunking results on the test set of the CoNLL 2000 dataset.

- CRF$^{\otimes}$(Flair)
  - This uses Flair embeddings [Akbik et al., 2018] with the CRF$^{\otimes}$. Akbik et al. [2018] recommend concatenating 100-dimensional GloVe embeddings to their forward and backward language model embeddings. The resulting Flair embeddings are 4196-dimensional.

Note that the GloVe embeddings are non-contextual whereas the BERT and Flair embeddings are contextual. Throughout training, we update the embeddings in the GloVe versions but do not update the BERT or Flair embedding models.

In early experiments, we also trained CRF$^{\otimes}$(BERT, biLSTM) and CRF$^{\otimes}$(Flair, biLSTM) models but found that these did not improve performance compared to using the CRF$^{\otimes}$ without the bidirectional LSTM. This is unsurprising because the BERT and Flair embeddings are already trained to encode sentence-level context.

In each model, for the feedforward networks $\mathbf{f}^{\phi}$, $\mathbf{f}^{\eta}$ and $\mathbf{f}^{\xi}$ referred to in Equations (4.9), (4.10) and (4.11), we use 2 layers with 600 units, ReLU activations and a skip connection. We train each model using stochastic gradient descent with a learning rate of 0.001 and Nesterov momentum of 0.9 [Nesterov, 1983]. We train for a maximum of 100,000 iterations, using early stopping on the validation set.

4.6.2 Results

The results for the four datasets are shown in Tables 4.2, 4.3, 4.4, and 4.5. Across all of the tasks, our results are competitive with the best published methods. We find that the previous state of the art is outperformed by both CRF$^{\otimes}$(BERT) and

Model	Accuracy
CRF$^{\otimes}$(GloVe)	97.15
CRF$^{\otimes}$(GloVe, biLSTM)	97.15
CRF$^{\otimes}$(BERT)	97.24
CRF$^{\otimes}$(Flair)	97.56
Bohnet et al. [2018]	**97.96**
Akbik et al. [2018]	97.85
Ling et al. [2015]	97.78

Table 4.3: Part-of-speech tagging results on the test set of the Penn Treebank dataset.

Model	F1
CRF$^{\otimes}$(GloVe)	91.81
CRF$^{\otimes}$(GloVe, biLSTM)	91.34
CRF$^{\otimes}$(BERT)	93.82
CRF$^{\otimes}$(Flair)	94.22
Yamada et al. [2020]	**94.30**
Luoma and Pyysalo [2020]	93.74
Baevski et al. [2019]	93.50

Table 4.4: Named entity recognition results on the test set of the CoNLL 2003 dataset.

CRF$^{\otimes}$(Flair) on chunking on CoNLL 2000 and by CRF$^{\otimes}$(BERT) on named entity recognition on OntoNotes 5.0.

We find that on all of the tasks, the BERT and Flair versions outperform both of the GloVe versions. With CRF$^{\otimes}$(Glove, biLSTM), the bidirectional LSTM does encode sentence-level contextual information. However, it is trained on a much smaller amount of data than the BERT and Flair embedding models, likely resulting in its lower scores.

In general, there is little difference in performance between CRF$^{\otimes}$(Glove) and CRF$^{\otimes}$(Glove, biLSTM). This may suggest that the local context used by the CRF$^{\otimes}$ is sufficient for these tasks and that the wider context provided by the bidirectional LSTM does not add significant value when using the CRF$^{\otimes}$. This is consistent with the results provided in Section 4.9.3.

Model	F1
$\text{CRF}^{\otimes}$(GloVe)	88.70
$\text{CRF}^{\otimes}$(GloVe, biLSTM)	89.04
$\text{CRF}^{\otimes}$(BERT)	**92.17**
$\text{CRF}^{\otimes}$(Flair)	91.54
Li et al. [2020b]	92.07
Yu et al. [2020]	91.30
Li et al. [2020a]	91.11

Table 4.5: Named entity recognition results on the test set of the OntoNotes 5.0 dataset.

4.7 Ablation study

In this section, we attempt to understand the effects of the local context and nonlinear potentials when compared to the CRF.

To this end, we train three additional variants of each of the model versions described in Section 4.6.1. In each case, we replace the $\text{CRF}^{\otimes}$ component with one of the following:

- $\text{CRF}^{\times}$
    - This is the same as the $\text{CRF}^{\otimes}$, but with linear (instead of nonlinear) potential functions.
- $\text{CRF}^{\bigcirc}$
    - This removes the local context, i.e. it is the same as the $\text{CRF}^{\otimes}$ but with the ϕ_l and ξ_l potentials removed. The η_l potential function is still nonlinear.
- CRF
    - This is the linear chain CRF. It is the same as the $\text{CRF}^{\bigcirc}$, but the η_l potential function is linear instead of nonlinear.

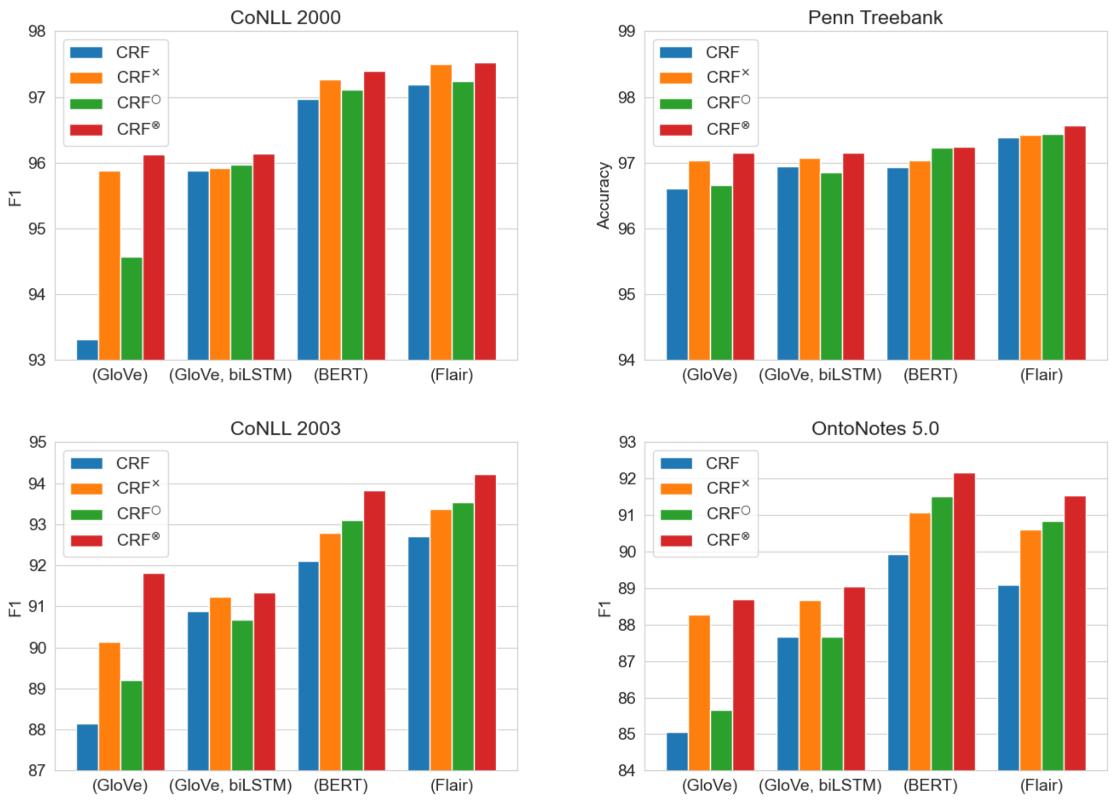


Figure 4.4: Results of the ablation study on each of the test sets. We compare the results when augmenting the CRF with locally-contextual connections, nonlinear potential functions, and both.

4.7.1 Results

Figure 4.4 shows the results with each of these variants. In all cases, we see that the local context is helpful: for each set of embeddings, the results for the CRF$^{\otimes}$ are higher than for the CRF$^{\bigcirc}$ and the results for the CRF$^{\times}$ are higher than for the CRF. The story is similar for the nonlinear potentials; in almost all cases, the CRF$^{\otimes}$ is better than the CRF$^{\times}$ and the CRF$^{\bigcirc}$ is better than the CRF.

Table 4.6 shows example sentences from the two named entity recognition test sets where CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) make correct predictions but CRF(BERT) and CRF(Flair) do not. In the first example, CRF(BERT) mistakes the length of the organisation “St Pius X High School”. This does not happen with CRF$^{\otimes}$(BERT), most likely because the model is aware that the word after “High” is “School” and therefore that this is a single organisation entity. Similarly, in the second example both CRF(BERT) and CRF(Flair) struggle to identify whether or not the quantities are cardinals or if they are related to the time measurements. In contrast, CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) do not suffer from the same mistakes.

While the CRF$^{\otimes}$ models work well, it is interesting to examine where they still make errors. Table 4.7 shows example sentences from the named entity recognition test sets where CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) make incorrect predictions. We see that in both examples, both models correctly identify the positions of the named entities but struggle with their types. The first example is one that even a human may struggle to label without any external knowledge beyond the sentence itself. In the second example, both models fail to recognise the one long entity and instead break it down into several smaller, plausible entities.

4.7.2 Computational efficiency

Table 4.8 in Section 4.9.1 shows the detailed computational efficiency statistics of each of the models we train. In summary, we find that the CRF$^{\otimes}$ takes approximately 1.5 to 2 times longer for each training and inference iteration than the CRF. The CRF$^{\times}$ and CRF$^{\otimes}$ models have approximately 3 times as many parameters as the CRF and CRF$^{\bigcirc}$ respectively.

4.8 Conclusion

We propose locally-contextual nonlinear CRFs for sequence labelling. Our approach directly incorporates information from the neighbouring embeddings when predicting the label for a given word, and parametrises the potential functions using deep neural networks. Our model serves as a drop-in replacement for the linear chain CRF, consistently outperforming it in our ablation study. On a variety of tasks, our results are competitive with those of the best published methods. In particular, we outperform the previous state of the art on chunking on CoNLL 2000 and named entity recognition on OntoNotes 5.0 English.

Compared to the CRF, the CRF$^{\otimes}$ makes it easier to use locally-contextual information when predicting the label for a given word in a sentence. However if contextual embedding models such as BERT and Flair were jointly trained with

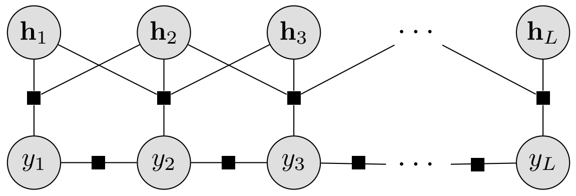


Figure 4.5: The graphical model of the CRF$^\otimes$-Concat.

sequence labelling tasks, this may eliminate the need for the additional locally-contextual connections of the CRF$^\otimes$; investigating this would be a particularly interesting avenue for future work.

The example sentences discussed in Section 4.7 showed that the CRF$^\otimes$ model could benefit from being combined with a source of external knowledge (for example, the knowledge that "Busang" is a location in the first example in Table 4.7). Therefore another fruitful direction for future work could be to augment the CRF$^\otimes$ with a knowledge base. This has been shown to improve performance on named entity recognition in particular [Kazama and Torisawa, 2007, Seyler et al., 2017, He et al., 2020].

4.9 Appendix

4.9.1 Computational efficiency

Table 4.8 shows the detailed computational efficiency statistics of each of the models we train. For a given embedding model, we find that the CRF$^\otimes$ takes approximately 1.5 to 2 times longer for each training and inference iteration than the CRF. When not using the bidirectional LSTM, the CRF$^\times$ and CRF$^\otimes$ have approximately 3 times as many parameters as the CRF and CRF$^\bigcirc$ respectively. However when using the bidirectional LSTM, this component dominates the number of parameters. This means that the CRF$^\times$(GloVe, biLSTM) has approximately the same number of parameters as the CRF(GloVe, biLSTM) and the CRF$^\otimes$(GloVe, biLSTM) only has approximately 1.6 times as many parameters as the CRF$^\bigcirc$(GloVe, biLSTM).

4.9.2 A more general parametrisation

A more general parametrisation of the local context would, at each time step, have a single potential for $\mathbf{h}_{l-1}$, $\mathbf{h}_l$, $\mathbf{h}_{l+1}$ and y_l instead of three separate ones as in Section 4.3.

The graph of this model, which we refer to as CRF$^{\otimes}$-Concat, is shown in Figure 4.5. We define the potential

$$\sigma_l = \sigma(y_l, \mathbf{h}_{l-1}, \mathbf{h}_l, \mathbf{h}_{l+1}; \theta). \tag{4.17}$$

Then, the model is parametrised as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \frac{\prod_l \psi_l \sigma_l}{\sum_{\mathbf{y}} \prod_l \psi_l \sigma_l}. \tag{4.18}$$

The parametrisation of $\log \psi_l$ remains the same as in Equation (4.4). We parametrise $\log \sigma_l$ as

$$\log \sigma_l = \mathbf{i}(y_l)^{\mathsf{T}} \mathbf{f}^{\sigma}([\mathbf{h}_{l-1}; \mathbf{h}_l; \mathbf{h}_{l+1}]) \tag{4.19}$$

where $\mathbf{f}^{\sigma}$ is a feedforward network which takes as input the concatenated embeddings $\mathbf{h}_{l-1}$, $\mathbf{h}_l$, and $\mathbf{h}_{l+1}$.

We train this model with BERT and Flair embeddings. As per Section 4.6.1, $\mathbf{f}^{\sigma}$ has 2 layers with 600 units, ReLU activations and a skip connection. This means that for any choice of word embeddings, CRF$^{\otimes}$ and CRF$^{\otimes}$-Concat have approximately the same number of parameters.

The results are shown in Table 4.9. We see that in all cases, CRF$^{\otimes}$ performs better than CRF$^{\otimes}$-Concat. This result is likely due to CRF$^{\otimes}$ being easier to train than CRF$^{\otimes}$-Concat: as well as having better test set scores, we find that CRF$^{\otimes}$ consistently achieves a higher value for the training objective than CRF$^{\otimes}$-Concat, which suggests that it finds a better local optimum.

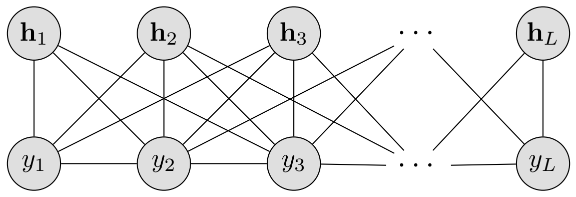


Figure 4.6: The graphical model of the CRF$^{\otimes}$-Wide.

4.9.3 Widening the contextual window

The CRF$^{\otimes}$ as described in Section 4.3 has a local context of width 3. That is, the embeddings $\mathbf{h}_{l-1}$, $\mathbf{h}_l$ and $\mathbf{h}_{l+1}$ are directly used when modelling the label y_l. One may consider that the performance of the model would be better with a wider context.

The graph of this model, which we refer to as CRF$^{\otimes}$-Wide, is shown in Figure 4.6. In addition to the potentials in Equations (4.2), (4.3), (4.7) and (4.8), we define

$$\pi_l = \pi(y_l, \mathbf{h}_{l-2}; \theta) \tag{4.20}$$

$$\zeta_l = \zeta(y_l, \mathbf{h}_{l+2}; \theta). \tag{4.21}$$

The model is then parametrised as

$$p(\mathbf{y}|\mathbf{x}, \theta) = \frac{\prod_l \psi_l \pi_l \phi_l \eta_l \xi_l \zeta_l}{\sum_{\mathbf{y}} \prod_l \psi_l \pi_l \phi_l \eta_l \xi_l \zeta_l}. \tag{4.22}$$

As with the parametrisation of the potentials ϕ_l, η_l and ξ_l in Equations (4.9) to (4.11), $\log \pi_l$ and $\log \zeta_l$ are parametrised using feedforward networks $\mathbf{f}^{\pi}$ and $\mathbf{f}^{\zeta}$ whose inputs are the embeddings $\mathbf{h}_{l-2}$ and $\mathbf{h}_{l+2}$ respectively as

$$\log \pi_l = \mathbf{i}(y_l)^{\mathsf{T}} \mathbf{f}^{\pi}(\mathbf{h}_{l-2}) \tag{4.23}$$

$$\log \zeta_l = \mathbf{i}(y_l)^{\mathsf{T}} \mathbf{f}^{\zeta}(\mathbf{h}_{l+2}). \tag{4.24}$$

We train this model with BERT and Flair embeddings. As per Section 4.6.1, $\mathbf{f}^{\pi}$ and $\mathbf{f}^{\zeta}$ each have 2 layers with 600 units, ReLU activations and a skip connection.

The results are shown in Table 4.10. In general, we see that there is very little difference in performance between CRF$^{\otimes}$ and CRF$^{\otimes}$-Wide. This suggests that the local context used by the CRF$^{\otimes}$ is sufficiently wide for the tasks evaluated.

Dataset	Model	Prediction
CoNLL 2003	Ground truth & CRF$^{\otimes}$(BERT) & CRF$^{\otimes}$(Flair)	[***PER*** Mike Cito], 17, was expelled from [***ORG*** St Pius X High School] in [***LOC*** Albuquerque] after an October game in which he used the sharpened chin strap buckles to injure two opposing players and the referee.
	CRF(BERT)	[***PER*** Mike Cito], 17, was expelled from [***ORG*** St Pius X] High School in [***LOC*** Albuquerque] after an October game in which he used the sharpened chin strap buckles to injure two opposing players and the referee.
	CRF(Flair)	[***PER*** Mike Cito], 17, was expelled from [***LOC*** St Pius X High School] in [***LOC*** Albuquerque] after an October game in which he used the sharpened chin strap buckles to injure two opposing players and the referee.
OntoNotes 5.0	Ground truth & CRF$^{\otimes}$(BERT) & CRF$^{\otimes}$(Flair)	It is [***TIME*** three hours] by car to [***GPE*** Hong Kong] and [***TIME*** one and a half hours] by boat to [***GPE*** Humen].
	CRF(BERT)	It is [***CARDINAL*** three] hours by car to [***GPE*** Hong Kong] and [***TIME*** one and a half hours] by boat to [***GPE*** Humen].
	CRF(Flair)	It is [***TIME*** three hours] by car to [***GPE*** Hong Kong] and [***CARDINAL*** one] and [***TIME*** a half hours] by boat to [***GPE*** Humen].

Table 4.6: Example sentences from the named entity recognition test sets where the CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) models make the correct predictions but the CRF(BERT) and CRF(Flair) models make the incorrect predictions.

Dataset	Model	Prediction
CoNLL 2003	Ground truth	[***ORG*** Bre-X], [***ORG*** Barrick] said to continue [***LOC*** Busang] talks.
	CRF$^{\otimes}$(BERT)	[***ORG*** Bre-X], [***PER*** Barrick] said to continue [***ORG*** Busang] talks.
	CRF$^{\otimes}$(Flair)	[***LOC*** Bre-X], [***ORG*** Barrick] said to continue [***MISC*** Busang] talks.
OntoNotes 5.0	Ground truth	In the near future, [***EVENT*** the Russian Tumen River Region Negotiation Conference] will also be held in [***GPE*** Vladivostok].
	CRF$^{\otimes}$(BERT)	In the near future, the [***NORP*** Russian] [***LOC*** Tumen River] [***ORG*** Region Negotiation Conference] will also be held in [***GPE*** Vladivostok].
	CRF$^{\otimes}$(Flair)	In the near future, the [***NORP*** Russian] [***LOC*** Tumen River Region] [***EVENT*** Negotiation Conference] will also be held in [***GPE*** Vladivostok].

Table 4.7: Example sentences from the named entity recognition test sets where the CRF$^{\otimes}$(BERT) and CRF$^{\otimes}$(Flair) models make the incorrect predictions.

Model	CoNLL 2000			Penn Treebank		
	Prms	Trn	Inf	Prms	Trn	Inf
CRF(GloVe)	<0.1	0.05	0.08	<0.1	0.07	0.10
CRF$^{\times}$(GloVe)	<0.1	0.07	0.10	<0.1	0.10	0.14
CRF$^{\bigcirc}$(GloVe)	0.7	0.05	0.09	0.8	0.08	0.13
CRF$^{\otimes}$(GloVe)	2.2	0.10	0.15	2.3	0.13	0.18
CRF(GloVe, biLSTM)	1.6	0.07	0.10	1.6	0.09	0.13
CRF$^{\times}$(GloVe, biLSTM)	1.6	0.11	0.15	1.7	0.13	0.18
CRF$^{\bigcirc}$(GloVe, biLSTM)	2.4	0.07	0.12	2.4	0.11	0.16
CRF$^{\otimes}$(GloVe, biLSTM)	3.8	0.12	0.18	3.9	0.16	0.22
CRF(BERT)	<0.1	0.05	0.09	<0.1	0.12	0.17
CRF$^{\times}$(BERT)	0.1	0.09	0.14	0.1	0.14	0.21
CRF$^{\bigcirc}$(BERT)	1.3	0.07	0.11	1.3	0.15	0.20
CRF$^{\otimes}$(BERT)	3.9	0.11	0.18	4.0	0.22	0.29
CRF(Flair)	0.1	0.20	0.27	0.2	0.36	0.42
CRF$^{\times}$(Flair)	0.3	0.29	0.39	0.6	0.38	0.43
CRF$^{\bigcirc}$(Flair)	5.5	0.25	0.35	5.6	0.36	0.45
CRF$^{\otimes}$(Flair)	16.5	0.35	0.48	16.9	0.52	0.63

Model	CoNLL 2003			OntoNotes 5.0		
	Prms	Trn	Inf	Prms	Trn	Inf
CRF(GloVe)	<0.1	0.07	0.11	<0.1	0.10	0.13
CRF$^{\times}$(GloVe)	<0.1	0.11	0.16	<0.1	0.14	0.18
CRF$^{\bigcirc}$(GloVe)	0.7	0.09	0.14	0.8	0.10	0.15
CRF$^{\otimes}$(GloVe)	2.2	0.16	0.20	2.3	0.17	0.22
CRF(GloVe, biLSTM)	1.6	0.11	0.16	1.6	0.13	0.17
CRF$^{\times}$(GloVe, biLSTM)	1.6	0.14	0.19	1.7	0.16	0.22
CRF$^{\bigcirc}$(GloVe, biLSTM)	2.4	0.13	0.17	2.4	0.13	0.18
CRF$^{\otimes}$(GloVe, biLSTM)	3.8	0.16	0.23	3.9	0.20	0.28
CRF(BERT)	<0.1	0.13	0.19	<0.1	0.18	0.23
CRF$^{\times}$(BERT)	<0.1	0.16	0.23	0.1	0.18	0.26
CRF$^{\bigcirc}$(BERT)	1.3	0.15	0.21	1.3	0.18	0.25
CRF$^{\otimes}$(BERT)	3.9	0.22	0.31	4.0	0.23	0.33
CRF(Flair)	<0.1	0.46	0.58	0.2	0.50	0.64
CRF$^{\times}$(Flair)	0.1	0.55	0.65	0.5	0.56	0.67
CRF$^{\bigcirc}$(Flair)	5.4	0.46	0.60	5.6	0.54	0.66
CRF$^{\otimes}$(Flair)	16.3	0.62	0.74	16.7	0.68	0.81

Table 4.8: Computational efficiency statistics of each the models trained. The Prms column shows the number of parameters, in millions. Trn shows the average time taken, in seconds, for a training iteration with a batch of 128 sentences. Inf shows the average time taken, in seconds, to infer the most likely labels for a batch of 256 sentences. All values were computed using the same GPU.

Model	CoNLL 2000 F1	Penn Treebank Accuracy
$\text{CRF}^{\otimes}$(BERT)	97.40	97.24
$\text{CRF}^{\otimes}$-Concat(BERT)	97.23	97.10
$\text{CRF}^{\otimes}$(Flair)	97.52	97.56
$\text{CRF}^{\otimes}$-Concat(Flair)	97.48	97.44

Model	CoNLL 2003 F1	OntoNotes 5.0 F1
$\text{CRF}^{\otimes}$(BERT)	93.82	92.17
$\text{CRF}^{\otimes}$-Concat(BERT)	93.10	91.80
$\text{CRF}^{\otimes}$(Flair)	94.22	91.54
$\text{CRF}^{\otimes}$-Concat(Flair)	93.90	91.30

Table 4.9: Test set results with a more general parametrisation of the local context.

Model	CoNLL 2000 F1	Penn Treebank Accuracy
$\text{CRF}^{\otimes}$(BERT)	97.40	97.24
$\text{CRF}^{\otimes}$-Wide(BERT)	97.39	97.19
$\text{CRF}^{\otimes}$(Flair)	97.52	97.56
$\text{CRF}^{\otimes}$-Wide(Flair)	97.50	97.53

Model	CoNLL 2003 F1	OntoNotes 5.0 F1
$\text{CRF}^{\otimes}$(BERT)	93.82	92.17
$\text{CRF}^{\otimes}$-Wide(BERT)	93.81	92.15
$\text{CRF}^{\otimes}$(Flair)	94.22	91.54
$\text{CRF}^{\otimes}$-Wide(Flair)	94.23	91.44

Table 4.10: Test set results with a wider local context.

Conclusion

Generative models aim to simulate the process by which a set of data is generated. They are intuitive, interpretable and are naturally suited to learning from unlabelled data. Traditional generative modelling approaches have achieved success on several natural language processing tasks, however they can often be inflexible due to the need to maintain tractable maximum likelihood training. In contrast, deep learning methods are powerful and flexible, and are very popular in modern natural language processing. In recent years, algorithms have been developed for training generative models that incorporate deep learning. In this work, we have shown several ways to leverage such algorithms to develop intuitive and powerful deep generative models for natural language which achieve competitive performance on a variety of tasks.

In Chapter 2 we presented a latent variable model to learn representations of sentences in an unsupervised manner. Prior deep generative modelling approaches to this task were typically based on recurrent neural networks. However they suffered from the KL collapse phenomenon whereby the approximate posterior distribution would not learn an informative representation of a sequence of text. Instead, inspired by a successful approach from image modelling, we presented a model which uses a dynamic attention mechanism to iteratively update a canvas that parametrises the probability distribution over the sentence's text. We found that this method alleviates the KL collapse issue suffered by prior approaches, allowing for useful semantic representations to be inferred. As a result, the model is able to generate diverse, coherent sentences, to successfully impute missing words, and to find semantically similar sentences.

We then hypothesised that using sentences expressed in multiple languages allows for richer representations to be learned than from a single language alone.

Therefore in Chapter 3 we presented a model for machine translation which uses a latent variable as a language-agnostic representation that is encouraged to learn the semantic meaning of the sentence. We used this model to perform multilingual translation, and leverage monolingual sentences during training. This approach achieves competitive BLEU scores (particularly when the amount of paired training data is limited) and is especially effective at translating long sentences. When there are missing words in the source sentence, the model is able to use its learned representation to infer what those words may be and produce good translations accordingly. This work has subsequently been built upon in various interesting ways. For example, Eikema and Aziz [2019] use our model but generate translations using a non-iterative procedure based on an approximation to the inference distribution. Zheng et al. [2020] modify our graphical model by using a mirrored structure in order to directly model bidirectional translation. Zhang et al. [2019] adapt our model for jointly modelling sentences and their syntactic trees.

In recent years, deterministic contextual token embeddings provided by pre-trained language models have become more and more popular. However, these language models are trained in a manner that is agnostic to the downstream task they will eventually be used for. It can therefore be necessary to design the architecture for the downstream task to directly extract the most useful information from the embeddings. To this end, in Chapter 4 we presented a locally-contextual CRF for sequence labelling tasks. This approach enhanced the CRF by directly incorporating the neighbouring words when predicting the label for a given word and by using deep, nonlinear potential functions. We found improved results on several sequence labelling tasks, and showed that both the local context and nonlinear potentials consistently provide improvements compared to CRFs.

This dissertation presents several potential avenues for future work:

Learning semantic representations We have not fully evaluated the limits of the representations learned in this thesis. Do they really ‘understand’ the meaning of the sentences they represent? How can we measure this? In this direction, there is straightforward future work which involves assessing the performance of these representations on tasks which test their ability to understand text.

One may also ask the question: are the proposed tasks optimal for learning representations which understand sequences of text? For example, we have recently seen significant performance gains from methods which use self-supervised contrastive learning. Could this, or other paradigms, be better ways to train our models?

Learning at scale In the past few years, contextual embedding methods (e.g. BERT, Flair, etc.) have become extremely popular. By using the outputs of these pre-trained models, state of the art results have been achieved on a wide range of tasks. These methods typically operate at a very large scale, both in terms of model size and the amount of training data. The latent variable models proposed in this work have certain inherent benefits, but have few parameters and have been trained on relatively small amounts of data. Can we realise the benefits of latent variable models at scale? How will we address the various challenges that training large such models is likely to bring about (e.g. instability, KL collapse, etc.)?

Multi-task learning For a model to truly be considered ‘intelligent’, it will likely have to be able to perform multiple tasks with data from multiple modalities. In the future, we hope to develop such a model using some of the techniques from this work, particularly those of Chapter 3. This will present a number of challenges (including parameter sharing, overcoming catastrophic forgetting, dealing with different data types, etc.) which will need to be addressed.

Advances in generative modelling The models presented throughout this thesis are arguably very simple in terms of their graphical structures, the functional forms used to parametrise the various distributions, and the maximum likelihood objectives used to train them. Although this can be beneficial (e.g. in terms of intuitiveness and interpretability), it raises the question: does the simplicity of these models limit how well they perform in practice? In recent years, there have been significant advances in generative modelling both in terms of how to design models as well as how to train them (e.g. flow-based models, energy-based models, diffusion models, etc.). How can we adopt these methods and apply them to the models/tasks in this work?

Bibliography

A. Akbik, D. Blythe, and R. Vollgraf. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, 2018.

A. Baevski, S. Edunov, Y. Liu, L. Zettlemoyer, and M. Auli. Cloze-driven Pretraining of Self-attention Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2019.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, 2015.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3, 2003.

Y. Bengio, A. Courville, and P. Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35, 2013.

K. Bhattacharjee, M. Ballesteros, R. Anubhai, S. Muresan, J. Ma, F. Ladhak, and Y. Al-Onaizan. To BERT or Not to BERT: Comparing Task-specific and Task-agnostic Semi-Supervised Approaches for Sequence Tagging. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing EMNLP*, 2020.

D. M. Bikel, R. Schwartz, and R. M. Weischedel. An Algorithm That Learns What‘s in a Name. *Machine Learning*, 34, 1999.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3, 2003.

B. Bohnet, R. McDonald, G. Simões, D. Andor, E. Pitler, and J. Maynez. Morphosyntactic Tagging with a Meta-BiLSTM Model over Context Sensitive Token Encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance Weighted Autoencoders. In *International Conference on Learning Representations*, 2016.

K. Clark, M.-T. Luong, C. D. Manning, and Q. Le. Semi-Supervised Sequence Modeling with Cross-View Training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2011.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39, 1977.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

A. B. Dieng, C. Wang, J. Gao, and J. Paisley. TopicRNN: A Recurrent Neural Network with Long-Range Semantic Dependency. In *International Conference on Learning Representations*, 2017.

B. Eikema and W. Aziz. Auto-Encoding Variational Neural Machine Translation. In *Proceedings of the 4th Workshop on Representation Learning for NLP*, 2019.

A. Eisele and Y. Chen. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, 2010.

X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning.* MIT Press, 2016.

A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks.* Springer, 2012.

A. Graves. Generating Sequences With Recurrent Neural Networks. *CoRR*, abs/1308.0850, 2014.

K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A Recurrent Neural Network For Image Generation. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Q. He, L. Wu, Y. Yin, and H. Cai. Knowledge-Graph Augmented Word Representations for Named Entity Recognition. In *Proceedings of the Thrity-Fourth AAAI Conference on Artificial Intelligence*, 2020.

S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9, 1997.

K. Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4, 1991.

Z. Hu, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu. An Investigation of Potential Function Designs for Neural CRF. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020.

Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR*, abs/1508.01991, 2015.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *Transactions of the Association for Computational Linguistics*, 5, 2017.

J. Kazama and K. Torisawa. Exploiting Wikipedia as External Knowledge for Named Entity Recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007.

Y. Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

J. Kupiec. Robust Part-of-Speech Tagging Using a Hidden Markov Model. *Computer Speech & Language*, 6, 1992.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, 521, 2015.

X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li. A Unified MRC Framework for Named Entity Recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020a.

X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li. Dice Loss for Data-imbalanced NLP Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020b.

W. Ling, C. Dyer, A. W. Black, I. Trancoso, R. Fermandez, S. Amir, L. Marujo, and T. Luís. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

L. Liu, X. Ren, Q. Zhu, S. Zhi, H. Gui, H. Ji, and J. Han. Heterogeneous Supervision for Relation Extraction: A Representation Learning Approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

Y. Liu, F. Meng, J. Zhang, J. Xu, Y. Chen, and J. Zhou. GCDT: A Global Context Enhanced Deep Transition Architecture for Sequence Labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Y. Luo, F. Xiao, and H. Zhao. Hierarchical Contextualized Representation for Named Entity Recognition. In *Proceedings of the Thrity-Fourth AAAI Conference on Artificial Intelligence*, 2020.

J. Luoma and S. Pyysalo. Exploring Cross-sentence Contexts for Named Entity Recognition with BERT. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

X. Ma and E. Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19, 1993.

A. McCallum and W. Li. Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003.

T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *International Conference on Learning Representations*, 2013a.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, 2013b.

Y. E. Nesterov. A Method for Solving the Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269, 1983.

D. Q. Nguyen and K. Verspoor. An Improved Neural Network Model for Joint POS Tagging and Dependency Parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 2018.

J. W. Paisley, D. M. Blei, and M. I. Jordan. Variational Bayesian Inference with Stochastic Search. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

S. Park, S. Kwon, B. Kim, S. Han, H. Shim, and G. G. Lee. Question Answering System using Multiple Information Source and Open Type Answer Merge. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, 2015.

J. Peng, L. Bo, and J. Xu. Conditional Neural Fields. In *Advances in Neural Information Processing Systems*, 2009.

J. Pennington, R. Socher, and C. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

S. Pradhan, A. Moschitti, N. Xue, H. T. Ng, A. Björkelund, O. Uryupina, Y. Zhang, and Z. Zhong. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 2013.

L. Rabiner and B. Juang. An introduction to hidden Markov models. *IEEE Acoustics, Speech & Signal Processing Magazine*, 3, 1986.

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77, 1989.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI, 2018.

R. Reddy. Speech Understanding Systems: A Summary of Results of the Five-Year Research Effort. Technical report, Carnegie-Mellon University, 1977.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

J. Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61, 2015.

S. Semeniuta, A. Severyn, and E. Barth. A Hybrid Convolutional Variational Autoencoder for Text Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017.

R. Sennrich, B. Haddow, and A. Birch. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

D. Seyler, T. Dembelova, L. D. Corro, J. Hoffart, and G. Weikum. KnowNER: Incremental Multilingual Knowledge in Named Entity Recognition. *CoRR*, abs/1709.03544, 2017.

F. Sha and F. Pereira. Shallow Parsing with Conditional Random Fields. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003.

H. Shah and D. Barber. Generative Neural Machine Translation. In *Advances in Neural Information Processing Systems*, 2018.

H. Shah, B. Zheng, and D. Barber. Generating Sentences Using a Dynamic Canvas. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

H. Shah, T. Xiao, and D. Barber. Locally-Contextual Nonlinear CRFs for Sequence Labeling. *CoRR*, abs/2103.16210, 2021.

D. Shen, A. Celikyilmaz, Y. Zhang, L. Chen, X. Wang, J. Gao, and L. Carin. Towards Generating Long and Coherent Text with Multi-Level Latent Variable Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

R. Shu, H. H. Bui, and M. Ghavamzadeh. Bottleneck Conditional Density Estimation. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 2016.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems*, 2014.

J. Tiedemann. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, 2012.

E. F. Tjong Kim Sang and S. Buchholz. Introduction to the CoNLL-2000 Shared Task Chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000.

E. F. Tjong Kim Sang and F. De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*, 2003.

I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schöelkopf. Wasserstein Auto-Encoders. In *International Conference on Learning Representations*, 2018.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

J. Turian, L.-A. Ratinov, and Y. Bengio. Word Representations: A Simple and General Method for Semi-Supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 2010.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.

A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13, 1967.

I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

J. Yu, B. Bohnet, and M. Poesio. Named Entity Recognition as Dependency Parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. Variational Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

J. Zhang and C. Zong. Exploiting Source-side Monolingual Data in Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

X. Zhang, Y. Yang, S. Yuan, D. Shen, and L. Carin. Syntax-Infused Variational Autoencoder for Text Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Y. Zhang, Q. Liu, and L. Song. Sentence-State LSTM for Text Representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

Z. Zheng, H. Zhou, S. Huang, L. Li, X.-Y. Dai, and J. Chen. Mirror-Generative Neural Machine Translation. In *International Conference on Learning Representations*, 2020.

Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, 2015.