A Bi-directional Unified Model for Information Retrieval

Jagadeesh Gorla

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of the

University College London.

Department of Computer Science

University College London

June 5, 2015

I, Jagadeesh Gorla, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Jagadeesh Gorla

To everyone who supported me in the past 10 years!

Abstract

Relevance matching between two information objects such as a document and query or a user and product (e.g. movie) is an important problem in information retrieval systems. The most common and most successful way to approach this problem is by probabilistically modelling the relevance between information objects, and computing their relevance matching as the probability of relevance. The objective of a probabilistic relevance retrieval model is to compute the probability of relevance between a given information object pair using all the available information about the individual objects (e.g., document and query), the existing relevance information on both objects and all the information available on other information objects (other documents, queries in the collection and the relevance information on them).

The probabilistic retrieval models developed to date are not capable of utilising all available information due to the lack of a unified theory for relevance modelling. More than three decades ago, the notion of simultaneously utilising the *relevance* information about individual user needs *and* individual documents to come to a retrieval decision was formalised as the problem of a *unified relevance model* for Information Retrieval (IR). Since the inception of the unified model, a number of unsuccessful attempts have been made to develop a formal probabilistic relevance model to solve the problem. This thesis provides a new theory and a probabilistic relevance framework that not only solves the problem of the original *unified relevance model* but also provides the capability to utilise any available information about the information objects in computing the probability of relevance.

In this thesis, we consider information matching between two objects (e.g. documents and queries) to be *bi-directional preference matching* and the relevance between them is thus established and estimated on top of the bi-directional relationship. A key benefit of this bi-directional approach is that the resulting probabilistic bi-directional unified model not only solves the original problem of a unified model in information retrieval but also has the ability to incorporate all of the available information on the information objects (documents and queries) into a single model while computing the probability of relevance. Theoretically, we demonstrate the effectiveness of applying our single framework by deriving relevance ranking functions for popular retrieval scenarios such as collaborative filtering (recommendation), group recommendation and ad-hoc retrieval. In the past, the solution for relevance matching in each of these retrieval scenarios approached with a different solution/framework, partly due to the kind of information available to the retrieval system for computing the probability of relevance. However, the underlying problem of information matching is the same in all scenarios, and a solution to the problem of a unified model should be applicable to all scenarios.

One of the interesting aspects of our new theory and model in applying to a collaborative filtering scenario is that it computes the probability of relevance between a given user and a given item while not applying any dimensionality reduction technique or computing the explicit similarity between the users/items, which is contrary to the state-of-the-art collaborative filtering/recommender models (e.g. Matrix Factorisation methods, neighbourhood-based methods). This property allows the retrieval model to model users and items independently with their own features, rather than forcing it to use a common feature space (e.g., common hidden factor-features between a user-item pair of objects or a common vocabulary space between a document-query pair of objects).

The effectiveness of this theoretical framework is demonstrated in various real-world applications by experimenting on datasets in collaborative filtering, group recommendation and ad-hoc retrieval tasks. For collaborative filtering and group recommendation the model convincingly out-performs various state-of-the-art recommender models (or frameworks). For ad-hoc retrieval, the model also outperforms the state-of-the-art information retrieval models when it is restricted to use the same information used by the other models. The bi-directional unified model allows the building of both search and personalisation/recommender (or collaborative filtering) systems from a single model, which has not been possible before with the existing probabilistic relevance models. Finally, our theory and its framework have been adopted by some large companies in gaming, venture-capital matching, retail and media, and deployed on their web systems to match their customers, often in the tens of millions, with relevant content.

Acknowledgements

I would like to thank my advisor, Dr. Jun Wang, for providing me with a great opportunity to pursue my curiosity. Since the beginning of my Ph.D., he not only provided me with his technical expertise but also helped me in other aspects. When I started my Ph.D., I had very little understanding of Information Retrieval. This thesis would not have been possible without the constant help and guidance from my advisor Prof. Stephen Robertson. His influence on me is beyond research! I am extremely grateful to him for all his great teachings.

It was a pleasure working with Ulrich Paquet (Microsoft Research) and Neal Lathia (Cambridge University). My stay at UCL would not have been so much fun without great colleagues and friends, especially Dr. Tamas Jambor, Dr. Hersh Asthana and Dr. Vasilios! Finally, I would like to thank Microsoft Research and EPSRC for their generous Ph.D. scholarship and conference travel gifts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information Retrieval (IR) is mainly concerned with searching for *relevant* information in large structured or unstructured collections. The concept of relevance is therefore important. In the past, many authors have attempted to define various aspects of relevance [Miz97] and many different models, both non-probabilistic and probabilistic, have been proposed to capture the notion of relevance. In general, *relevance* can be interpreted as how well the retrieved information item satisfies the information need of the user. Despite the concept of relevance seeming intuitive, it is hard to define and even harder to model in a formal fashion. Moreover, whether a user judges the given document for his information need as relevant or not involves a great number of variables about the document (e.g., the content, when it was written and the audience intended for the document) and his information need (e.g., his knowledge, motivation) [RMC82a]. Therefore, it is almost impossible to predict the exact relevance relationship between a query representing a user's information need and a given document. As such, the uncertainty in relevance is often modelled probabilistically for retrieving relevant documents in the Robertson Spärck Jones model (RSJ) and the language models [RSJ76, PC98], for example. In all the probabilistic retrieval models, the central problem is representing a given information need and an item, and estimating the probability of *relevance* between the information need and item (query-document) pair.

The traditional probabilistic retrieval models such as RSJ [RSJ76] and the language models [PC98] rely on bags of terms for representation. That is, the information need and the information item are represented (or indexed) with a set of terms in the vocabulary. The probability of relevance is computed by matching the terms used in the information need with the terms used in the item, along with previous relevance information on either the need or the item but not *both* [Rob03]. Whereas in probabilistic inference based logical models of information retrieval [vR86, WY91], the information need and the item are considered as logical propositions over a conceptual knowledge space. In general, each concept in the knowledge space corresponds to a term in the vocabulary, and the task of retrieval involves finding the probability of a document logically implying the query or a query implying the document or both implying each other. These probabilities of the document implying the query and the query implying the document are interpreted as measures of the exhaustivity and specificity of the document with respect to the query respectively. The exhaustivity indicates the topic coverage of the document with respect to the query terms and specificity represents the depth of topics covered by the document for the terms in the query [CK68, Jon72].

In both traditional and logical probabilistic relevance models, the probability of relevance is computed by representing both the need and the item in a common feature space of either vocabulary terms or a knowledge space of concepts corresponding to the terms. So, if the observed features of information need and the item are not from a common feature space then it is not straightforward to compute the probability of relevance between them using traditional or logical probabilistic relevance models. To better understand the problem, consider the traditional task of recommendation [DK04, WdVR08]. Recommender systems (e.g. for films) know about users, items and some history of user reactions to items, and try to recommend to each user those items he or she might be expected to like. One difference from the traditional retrieval task is that there is no assumption concerning the existence of features. In its simplest form, the recommendation task is assumed *only* to have past ratings, and no other information about either users or items. In this case, there are no common features between the user and item. This apart, the recommender task looks very much like the search task, particularly in terms of feedback or learning from the data. The recommendation of a particular item to a particular user should depend both on ratings that this user has given to other items (relevance information on user), and ratings that other users have given for this item (relevance information on item).

This last statement has been amply confirmed by previous research into recommender systems [DK04, WdVR08]. Initially many of the models investigated to address the problem of recommendation were either user-oriented (grouping similar items and basing recommendation on similarity between items rated by this user and those not yet seen), or item-oriented (grouping similar users and basing recommendation on items rated by similar users). Since then it has been found much better to combine both kinds of evidence [WdVR08]. Even though the problem is computing the probability of the relevance between the information need and item, as mentioned earlier, it is not straightforward to apply the traditional probabilistic relevance models as they do not have the ability to incorporate relevance information on both user and item. They also do not indicate how to use information from outside the common feature space between them (e.g., user age, location, item authority) [Rob03]. A simple such example of successful term weighting function with relevance is BM25 [RZ09]. BM25 incorporates the relevance information only on the query not on the document.

The usual (currently state-of-the-art) way of doing recommendation is to describe both users and items in terms of the same set of inferred/latent features, in a relatively low-dimensioned feature space [KBV09, Hof04]. One of the issues with this particular type of modelling is we need to map the observed native features, like terms describing the user/item, into the common latent feature space in order to compute the relevance between the user and the item. We can also discover other examples of relevance matching problems, such as serving ads from a webpage, dating agencies/services, and matching vacancies with job-seekers where we have multiple sources of information that can be used in computing the relevance between need and item. We simply do not have a single *probabilistic unified relevance retrieval model* that can be employed to solve these different matching problems.

This thesis presents a new information retrieval theory and a probabilistic unified retrieval model that address the above mentioned problems in computing the probability of relevance, namely

1. utilising the available relevance information on both the information need and the item
2. utilising information need specific and item specific information, and information from other similar needs and items
3. allowing the representation of both need and item with native features instead of representing in a latent feature space in all the matching scenarios with different sources of information.

In fact, the problem of computing the probability of relevance between a given need and item by utilising relevance information on both the need, the item and the information from other similar items and needs has been recognised as a problem for more than three decades. It was first formally defined by Robertson and his colleagues in 1980 and named it as the problem of a unified model for information retrieval [RMC82a]. In the following, we will provide a summary of the problem of the unified model as originally identified.

The unified model in information retrieval (IR) [RMC82a], which attempted to unify two prior probabilistic models of *relevance* [RSJ76, MK60a], presented a view of the information retrieval problem which has been the subject of several further modelling attempts, without (so far) a satisfactory full resolution [Rob03]. The original unified model in IR [RMC82a] was based on two prior probabilistic models. In the first [MK60a], called Model 1, a librarian indexing a book (an individual item, on the table in front of him/her) was invited to consider the kinds of user who might find this book interesting, and how these kinds of user would express their interest or request in terms of the subject headings available in the library system. The decision as to which heading(s) should be used to index this book should be based on the (librarian's estimate of) the probability that a user using any given heading would find the book interesting.

The second prior model [RSJ76], Model 2, is expressed from the point of view of an individual user making a request. If we have any known examples of items that this user likes, they can be used to characterise the kind of items that the user wants. New documents should then be selected and ranked based on the probability that they will satisfy the user's information need. When we combine the two models, as shown in Figure 1.1, we have a matrix of user needs (rows) against items (columns), with some notion of groups of similar user needs or similar items. The Model 1 probability of relevance considers a single column against multiple similar rows, while the Model 2 probability considers a single row against multiple similar columns.

The issue raised, which the unified model was an attempt to answer, is how to make use of information about the individual user need, *and simultaneously* of information about the individual document, to come to a retrieval decision. Figure 1.1 presents the different sources of information about the document and query that can be used in estimating the probability of relevance. The most obvious manifestation of this issue arises in the case of relevance feedback. We would like our systems to be capable of using relevance information (judgements) to improve search – both judgements made by this user in relation to the current need (which is traditional relevance feedback), and judgements made by other users on the

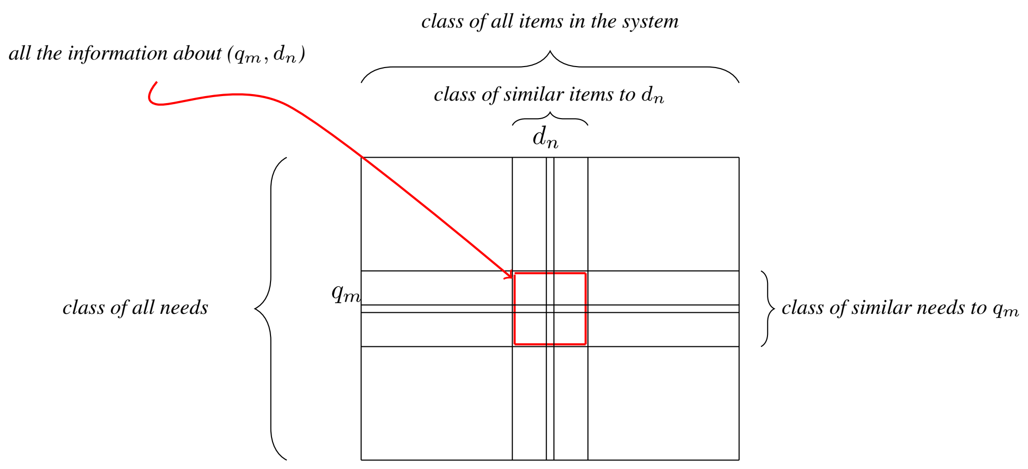

class of all items in the system
all the information about (q_m, d_n)
class of similar items to d_n
d_n
q_m
class of all needs
class of similar needs to q_m

Figure 1.1: The multiple sources of information that can be combined in estimating the probability of relevance between an information need and an item as depicted in the original unified model [RMC82a]. The q_m and d_n are the individual need and item respectively.

currently considered document. These latter judgements have become an important feedback/learning source in the web search environment.

Similar items or similar users are identified by properties or features of those items or users – in the Model 1 case, users are identified by what subject headings they use to express their interests, in the Model 2 case, items are identified by the words in them. Thus we have the Model 1 notion that we should index an item not by its own features but by the features of those users who will be interested in this item; and conversely, in Model 2, we should represent a user need not by its own features but by the features of the items that will be relevant to it. The contradiction identified in [RMC82b] lies exactly here. In (a simple version of) Model 1 we throw away the original item features in favour of applying learnt user-need features to the item; in Model 2 we throw away the user-need features in favour of applying learnt item features to the user need. If we throw away both sets of features, then we have nothing left on which to base anything. In the case of either Model 1 or Model 2, some kind of optimality is relatively straightforward to define (e.g. in Model 2, the optimal representation of a user need is that combination of item features which causes the best possible ranking of items, in terms of relevance to the user need). If we try to combine them, optimality is much more difficult to define in this particular way of modelling [Rob03].

So, this thesis presents a new theory that circumvents the above difficulties in finding a solution for the unified model, and provides a method to utilise the relevance information on both the need and the item, and the information from other similar needs and items, based on very simple and effective principles.

1.1 Contributions

The major contributions of the thesis are theoretical in nature. However, we validate our theory by providing some experimental evidence.

The contributions are the following:

1. We present a new theory for information retrieval under which information items (documents) and needs (queries) can be generated from two independent generative processes. We refer to our theory as the *theory of information matching*. Under our theory, the relevance matching between a given information item and need is computed as a function of *bi-directional preference matching* between the item and the need. This not only allows greater freedom in estimating the model parameters but also seamlessly incorporates item specific (e.g., *location*) or need specific features, and models relevance explicitly.

   The other traditional probabilistic retrieval model that defines two independent generative processes for document and query is the *risk-minimization framework* of Zhai and Lafferty's [LZ01]. However, the definition of relevance in this model is non-trivial. The authors circumvent this problem by removing the relevance from the model altogether and replacing it with a decision-theoretic notion of *risk*, which serves as the connection between two different generative processes. Moreover, the document and query generation feature space is a common feature space, e.g., vocabulary terms. If we have any feature that is not common to both query and document we cannot use it in computing relevance with this model.

2. Based on the above mentioned, we propose a formal bi-directional unified information matching framework (IMF) and a probabilistic relevance model for information retrieval, for computing the probability of relevance between two information objects (e.g., need and item). The model combines information from multiple sources of information and provides a single *symmetric unified relevance ranking function* for information matching, and can be applied to numerous problems including, collaborative filtering, people matching, group recommendation and ad-hoc retrieval.

   The resulting model computes the probability of relevance between a need-item pair without applying any dimensionality reduction techniques or computing any explicit similarity metric between the needs or items, in contrast to many state-of-the-art models, e.g. matrix factorisation and dimension reduction methods [KBV09, Hof04] and neighbourhood-based methods [SM95, SKKR01]. Note that the properties of the model are the same irrespective of the task we are performing.

   Our proposed model dynamically incorporates relevance feedback on both user and product (or document and query) while computing the relevance: that is, it is a true unified relevance retrieval model.

3. We present an approximate inference method based on Variational Bayes for predicting relevance in structured event-spaces and derive a symmetric probabilistic relevance ranking function. We provide relevance ranking solution for ad-hoc retrieval, collaborative filtering and group recommendation from the model. We evaluate the relevance ranking functions against various approaches. For ad-hoc retrieval, we present evaluation experiments by considering their effectiveness both with and without relevance judgements. We show how relevance information and

other sources of information are used by our unified model by using both example datasets and real-world datasets.

1.2 Related Publications, Reports and Industry applications

The following publications (and technical reports) are related to the thesis:

1. Jagadeesh Gorla, Neal Lathia, Stephen Robertson and Jun Wang: Probabilistic Group Recommendation via Information Matching. *In proceedings of the 22nd International World Wide Web Conference (WWW'13)*, Rio de Janeiro, Brazil, 2013.
2. Jagadeesh Gorla, Stephen Robertson, Jun Wang and Tamas Jambor: A Theory of Information Matching. *In ArXiv e-prints, CoRR*, 2012.
3. Jagadeesh Gorla, Stephen Robertson and Jun Wang: A Unified Relevance Retrieval Model by Eliteness Hypothesis. *In ArXiv e-prints, CoRR*, 2011.

The following large-scale web matching systems successfully deployed our framework to serve their customers (millions of users):

1. Relevant and Real-Time: Building a bi-directional recommendation system for a massive online game (RuneScape). *O'Reilly STRATA*, Barcelona, November 2014.
2. Match Capital, a matchmaking service for fundraising. *Financial Times*, August 2014.

1.3 Overview of the thesis

The remaining chapters of the thesis are organised in the following manner.

Chapter 2 will provide a general overview of the problem of unifying information in computing the probability of relevance and various successful formal probabilistic frameworks developed over past decades that are relevant to this thesis. We will start the chapter by introducing a brief introduction to the problem of relevance matching between two information objects. We then divide the relevant work on relevance matching problems into two popular retrieval tasks, namely text retrieval and recommendation (collaborative filtering and group recommendation). In section 2.1, we present various fundamental probabilistic frameworks for text retrieval including the Robertson-Spärck Jones (RSJ) model, language models, the probabilistic logical inference framework and the previous attempts at unified models. In section 2.2., we present well known frameworks in collaborative filtering, recommender systems and group recommendation frameworks.

In section 2.3, we provide a discussion on observations about various frameworks described in section 2.1 and 2.2 and present an overview of a generalised problem of information matching from the perspective of unifying the information from multiple sources in computing the probability of relevance. Finally, we draw conclusions and set the basis for Chapter 3.

Chapter 3 contains fundamental ideas of our thesis in developing a theory of information matching and unified information matching framework. In this chapter we will formally introduce the problem of information matching, present a theory of information matching and a unified information matching framework to solve the problem. We will also describe how this theory and framework can solve the problem of the unified relevance model of utilising all the available information on information objects in computing the probability of relevance. In section 3.1, we formally define the problem of information matching by defining different kinds of information that can be available to a retrieval system in computing the probability of relevance between two information objects. Section 3.2 will give an introduction to the formal elements of our new theory with the simple example of matching vacancies with job-seekers. Section 3.3 will discuss the unified information matching framework developed based on the theory of information matching presented in section 3.2. In section 3.4 we describe the mathematical formulation of our theory, framework and define the relevance function for computing the relevance between two information objects. Section 3.4 will give a probabilistic model of the theory and framework and provides its graphical model representation. Finally, we conclude in section 3.6 by providing a discussion on how our new theory and framework tackles the problems identified in previous models and provides a solution to the unified relevance model.

Chapter 4 takes the theory, unified information matching framework and the probabilistic model presented in Chapter 3 and shows how it can be applied to popular retrieval scenarios of collaborative filtering, group recommendation and ad-hoc text retrieval. In this chapter, we will discuss and derive a relevance ranking function for each retrieval scenario from our framework and show how different kinds of information on objects are being used when computing the probability of relevance between two information objects. In each case, we start by describing formal elements of each scenario and then map it into our framework to derive a relevance ranking function. We conclude each application by providing a discussion on its merits in comparison to the existing frameworks used to solve that particular retrieval scenario. We start by describing various scenarios of information matching and their corresponding popular retrieval scenarios. In section 4.1, we present an explanation to the mathematical function of the relevance under the bi-directional preference mappings developed for our new theory of information matching in Chapter 3. In section 4.2, 4.3 and 4.4, we present the application of the IMF to collaborative filtering, group recommendation and ad-hoc retrieval respectively. Section 4.5 will describe parameter estimation methods including a variational approximation method to perform inference in the information matching framework.

Chapter 5 is dedicated to experiments on applying the IMF to three different retrieval scenarios. We start by describing the objectives our experiments and the rationale behind choosing certain baseline frameworks/methods. In section 5.1, we present the results on applying the IMF to individual and group recommendation tasks. Section 5.2 will describe the results obtained by applying the IMF to ad-hoc retrieval task. In section 5.3, we will describe a preference search system developed using the IMF. Finally, we conclude the chapter in section 5.4.

Chapter 6 will conclude the thesis. We will discuss the future direction for the further development of our information matching framework.

Chapter 2

Related Work

In this chapter, we will present a brief introduction to the problem of the "unified model", a review of most relevant work and the importance of its solution.

The probabilistic modelling of relevance between two information objects is a common problem and goes beyond document retrieval. For instance, consider a simple example scenario of an online retailer with a product collection and a set of users with their own descriptions. An example of the features of a user *Jane* and a *shoe* product description are shown in Figure 2.1. Now, assume that the retailer would like to find a set of relevant products that a given user would like to buy/view, or a set of targeted (relevant) users for a given product. In our example, it is important for the retailer to find whether the given *shoe* product is relevant to *Jane* or not. In order to estimate the probability of *relevance* between the given product and user, we can use all available information about the product (e.g., colour, description), the user (e.g., age, location, about me), past behaviour of the user (what products *Jane* clicked, rated and purchased) and history of the product (who clicked, purchased and rated this particular product). In this context, this problem of estimating the probability of relevance can be defined as the problem of *matching* the most relevant products/users for a given user/product using all the available information. The structure of the problem is common in a number of domains including retrieval and recommendation. However, in practical systems, we do not always have access to all the above information. For example, in collaborative filtering [SM95, SKKR01], we only have interactions between users and products (ratings, clicks, etc.) but not the description of users/products. We may have both in commercially successful e-commerce or search systems, e.g., Amazon e-commerce company.

Similarly, in text retrieval, the fundamentals of the problem of estimating the probability of relevance matching are the same. In other words, in the simple ad-hoc retrieval task we have user query terms, and the documents described with the vocabulary terms. The probability can be estimated using the common attributes between document and query: vocabulary terms. In some cases, we may have relevance judgements on queries, documents, and an authority score of documents (e.g., PageRank). In this particular scenario, we have extra information about the query and the document (but not on the common attributes) that we can use for estimating the probability of relevance.

A probabilistic relevance model that can incorporate information from all available sources is called a *unified relevance retrieval model* [Rob03]. The fundamental objective of the thesis is to develop a

User id: u_1

Name: Jane Smith

Sex: Female

Age: 27

Location: Kensington

About Me: Business woman

Product id: p_1

Category: Shoe

Brand: Chanel

Colour: Red

Price: \$400

Description: Daily wear

Figure 2.1: An example of features from two distinct sets, describing different types of information object. Here, the user "Jane Smith" is represented with features such as age and location, whereas the product is represented with price and colour.

theory for a unified relevance retrieval model. One of the major difficulties of developing such a relevance model is defining a *probability space* that can accommodate events of different sources of information in computing the probability of relevance [Rob03].

In the following section we will give a description of different approaches used in estimating relevance to solve the matching problem in text retrieval and recommendation (both individual and group recommendation) that are relevant to the thesis. We present the fundamental principles and the ideas behind prominent models within each approach. In presenting the relevant work, we divide probabilistic retrieval models in text retrieval into five different categories or frameworks, namely

(a) probabilistic relevance framework (b) language modelling (c) logical inference modelling (d) probabilistic indexing (e) unified modelling

and, also divide the recommender models into three categories, namely

(a) collaborative filtering (b) content-based recommender systems (c) group recommendation

2.1 Text Retrieval

A number of probabilistic relevance models have been proposed to compute the probability of relevance in order to retrieve a set of relevant documents to a given user query. The fundamental principle behind these models is about *relevance* matching between two sets of properties, namely, the information need (query) and information item (document) properties that represent/describe need and item respectively. Often, the property space is common for both need and item, e.g. vocabulary terms. We note that documents and queries (needs) are typically represented by sets of properties – we may think of vocabulary terms for example.

2.1.1 Probabilistic Relevance Framework

The *Probabilistic Relevance Framework* (PRF) refers to a family of information retrieval models developed by Robertson and his colleagues over three decades [RZ09]. The PRF emphasises the importance of the *relevance* variable in the development of the retrieval models. There are two very successful models developed on PRF:

(a) The classical probabilistic model (or RSJ). (b) BM25 (a combination of RSJ and 2-Poisson model).

2.1.2 Classical Probabilistic Model (RSJ model)

One of the most influential probabilistic relevance models is the classical Probabilistic Model (or RSJ model) of Information Retrieval proposed by Robertson and Spärck Jones [RSJ76]. The fundamental principle behind the model is that it assumes that there is a set of relevant and another set of non-relevant documents for a given query. And, the model starts with a very general probability ranking principle [Rob97b], where the documents are ranked in the decreasing order of the probability of relevance to the given query, and derives the document relevance ranking score function.

Before we describe the estimation of the probability of relevance, we define the following notation. Let $\mathcal{D}$ and $\mathcal{Q}$ be the collection of documents and queries respectively. By d_m and q_n we denote the m^{th} document in $\mathcal{D}$ and the n^{th} query in $\mathcal{Q}$ respectively, where $m \in [1, |\mathcal{D}|]$ and $n \in [1, |\mathcal{Q}|]$. We represent every object of type $\mathcal{D}$ over a vector space of a set of $\mathcal{K}$ feature random variables and object $\mathcal{Q}$ over a vector space of a set of $\mathcal{L}$ feature random variables, i.e., $d_m \in \langle \boldsymbol{x_k} \rangle_{k=1}^{|\mathcal{K}|}$ and $q_n \in \langle \boldsymbol{y_l} \rangle_{l=1}^{|\mathcal{L}|}$. For example, in a simple bag of word model representation in our common notation, vocabulary terms are indexed into the set $\mathcal{K}$ (= $\mathcal{L}$) random variables and the document $d_m := \left\langle x_1^m, x_2^m \cdots, x_{|\mathcal{K}|}^m \right\rangle$ where x_k^m is a positive integer. The x_k^m is the term feature random variable value and it is either a binary presence/absence or the term frequency. Finally, we assume that relevance is binary and denote it by R ($R \in \{0, 1\}$).

The model derivation first transforms the probability of relevance into a likelihood ratio as follows,

$$P(R = 1|d_m, q_n) \propto_{q_n} \frac{P(R = 1|d_m, q_n)}{P(R = 0|d_m, q_n)} \tag{2.1}$$

By applying Bayesian transformation we get,

$$\begin{aligned} P(R = 1|d_m, q_n) &= \frac{P(d_m|R = 1, q_n)P(R = 1|q_n)}{P(d_m|R = 0, q_n)P(R = 0|q_n)} \\ &\propto_{q_n} \frac{P(d_m|R = 1, q_n)}{P(d_m|R = 0, q_n)} \end{aligned}$$

By assuming the independence between the document features, the above equation is rewritten as,

$$P(R = 1|d_m, q_n) \approx \prod_{k=1}^{|\mathcal{K}|} \frac{P(\boldsymbol{x_k} = x_k^m|R = 1, q_n)}{P(\boldsymbol{x_k} = x_k^m|R = 0, q_n)} \tag{2.2}$$

where $x_k^m \in \{0, 1\}$ representing whether the term corresponding to k^{th} feature is present or absent in d_m, where $d_m := \langle x_k^m \rangle_{k=1}^{|\mathcal{K}|}$.

We can further approximate the above equation by including only query specific terms and rewriting it as follows,

$$P(R=1|d_m,q_n) \approx \prod_{\exists k:(y_k^n=1)} \frac{P(\boldsymbol{x_k}=x_k^m|R=1)}{P(\boldsymbol{x_k}=x_k^m|R=0)} \tag{2.3}$$

where $q_n := \langle y_k^n \rangle_{k=1}^{|\mathcal{K}|}$. By applying log and separating the summation into the score for query terms that are present and absent in Eq. (2.3), we get

$$P(R=1|d_m,q_n) \approx \sum_{k:(y_k^n=1)} \left(log\frac{P(\boldsymbol{x_k}=1|R=1)}{P(\boldsymbol{x_k}=1|R=0)} + log\frac{P(\boldsymbol{x_k}=0|R=1)}{P(\boldsymbol{x_k}=0|R=0)} \right) \tag{2.4}$$

We can further rewrite Eq. (2.4) as

$$P(R=1|d_m,q_n) \approx \sum_{k:(y_k^n=1)} \left(log\frac{P(\boldsymbol{x_k}=1|R=1)\big(1-P(\boldsymbol{x_k}=1|R=0)\big)}{\big(1-P(\boldsymbol{x_k}=1|R=1)\big)P(\boldsymbol{x_k}=1|R=0)} \right), \tag{2.5}$$

see [RZ09].

Now, if we have a set of relevant and non-relevant document-query pairs, we can estimate all the four probability components of Eq. (2.5) using the documents frequency in the relevant and non-relevant sets with the presence of the k^{th} term. The same classical probabilistic model served as a fundamental model behind the later successful model BM25 [RZ09].

Observations:

1. The probabilities of relevance are estimated using relevance statistics on the query (i.e. based on the relevance of other documents to this query), but not on the document (i.e. not using the relevance of this document to other queries). So, the model cannot incorporate the relevance information on the document as well as the query.

2. The model can be used for relevance ranking if and only if both query and document share a common feature space (or projected into common feature space), e.g., vocabulary terms in text retrieval.

From the above observations, we conclude that the RSJ model is not designed to fulfil the requirements of the unified relevance retrieval model. However, we borrowed some of the fundamental principles behind the classical model in the process of developing a unified model in later chapters.

2.1.3 2-Poisson Eliteness Model and BM25

The classical probabilistic relevance model is based on the binary representations of the documents and queries, where a term is either present or not present in the document. In order to introduce the term frequency into the RSJ model, a new hidden property called *eliteness* is associated with a given document-term pair. In simple words, a term is *elite* to the document means that the document is *about* the concept represented by the term. Now, the assumption is that the term frequency of the term in a given document depends on its eliteness, and the relevance of the document to the query depends on the eliteness of query terms. Furthermore, the eliteness is assumed to be binary.

Under these assumptions, the term frequency tf of the k^{th} term in relation to the relevance can be written as follows,

$$P(\boldsymbol{x_k} = tf|R = 1) = P(\boldsymbol{x_k} = tf|E = 1)P(E = 1|R = 1) + P(\boldsymbol{x_k} = tf|E = 0)P(E = 0|R = 1)$$
$$P(\boldsymbol{x_k} = tf|R = 0) = P(\boldsymbol{x_k} = tf|E = 1)P(E = 1|R = 0) + P(\boldsymbol{x_k} = tf|E = 0)P(E = 0|R = 0)$$

where E is the binary eliteness property.

In order to estimate the probabilities $P(\boldsymbol{x_k} = tf|E = 1)$ and $P(\boldsymbol{x_k} = tf|E = 0)$, following Harter [Har75b], the model makes a distributional assumption that the term frequency of a term follows a Poisson distribution in an elite set of documents with mean μ_1 and another Poisson distribution in a non-elite set of documents with mean μ_0, and it is assumed that $\mu_1 >> \mu_0$. By substituting the distributional assumptions, we can write the above equations as follows,

$$P(\boldsymbol{x_k} = tf|R = 1) = \frac{e^{-\mu_1^k}(\mu_1^k)^{tf}}{tf!}P(E = 1|R = 1) + \frac{e^{-\mu_0^k}\mu_0^{k^{tf}}}{tf!}P(E = 0|R = 1)$$
$$P(\boldsymbol{x_k} = tf|R = 0) = \frac{e^{-\mu_1^k}(\mu_1^k)^{tf}}{tf!}P(E = 1|R = 0) + \frac{e^{-\mu_0^k}\mu_0^{k^{tf}}}{tf!}P(E = 0|R = 0)$$

By substituting the above equations into Eq.(2.3) we can compute the relevance score for a given query-document pair. In order to estimate the relevance ranking score using above equations, we need to estimate four different parameters, namely

(a) μ_1^k (b) μ_0^k (c) $P(E = 1|R = 1)$ (d) $P(E = 1|R = 0)$

The estimation of the above parameters is a daunting task. So, the authors approximated the relevance ranking weight of a query term k to a given document (d) with term frequency tf as

$$\overset{d}{w_k}(tf) = \frac{tf(k_1 + 1)}{k_1\big[1 - b + b\frac{dl}{avgDL}\big] + tf}\underbrace{\left(log\frac{(r_k + 0.5)(N - R - n(k) + r_k + 0.5)}{(n(k) - r_k + 0.5)(R - r_k + 0.5)}\right)}_{\text{RSJ weight}} \tag{2.6}$$

see [RZ09]. Where N is the total number of judged documents in the collection, R is the size of relevant set of documents, $n(k)$ is the number of documents in the judged sample with k^{th} vocabulary term, r_k is the number of judged relevant documents containing k^{th} term in the vocabulary and dl, $avgDL$ are document length ($|d|$) and average document length in the collection respectively. And b and k_1 are parameters where $b \in [0, 1]$ and k_1 is chosen such that $k_1 \in [1.2, 2.0]$. The weighting $\overset{d}{w_k}(tf)$ is referred to as the BM25 term weighting function, and has been highly successful.

The above BM25 weighting function uses the relevant information on queries in computing the relevance ranking term weight but it cannot incorporate the relevance information on the document. Thus, BM25 alone is a not a unified relevance retrieval model.

2.1.4 Language Models

The idea of using statistical language modelling for information retrieval was first proposed by Ponte & Croft in [PC98]. The main idea behind the language modelling approach is that every document d has an underlying language model θ_d. In basic language modelling approaches, there is no specific relevance variable except that the relevance is correlated with the process of generation of the query from the document. Therefore, a given query q is considered to be relevant to a document d if the query is a random sample from the model θ_d. Here, the task of an IR system is, first estimate the underlying language model θ_d of the document d and then calculate the probability of observing the query q_n as a random sample from θ_d (or probability of generating q from θ_d) and rank documents in the decreasing order of that probability.

Formally, let $V = \{t_1, t_2, \cdots t_k\}$ be the language vocabulary and the query is represented over a binary random vector over vocabulary space, i.e., $q = \langle x_1, x_2 \cdots x_k \rangle$ where $x_i = 1$ if i^{th} term is present in the query and 0 otherwise. In the basic language model [PC98], the probability of observing a query q from a given language model θ_d is computed as follows:

$$p(q|\theta_d) = \prod_{t_k \in q} p(x_k = 1|d) \prod_{t_k \notin q} (1 - p(x_k = 1|d)) \tag{2.7}$$

where the probability $p(x_k = 1|d)$ is estimated using maximum likelihood (ML) estimator as follows,

$$p(x_k = 1|d) = \frac{c(t_k, d)}{|d|} \tag{2.8}$$

where $c(t_k, d)$ is the count of term t_k in d and $|d|$ is the length of the document. If a term t_k does not occur in the document, the model uses its relative frequency in the entire collection, also referred as collection frequency. The Eq. (2.7) represents the multiple-Bernoulli distribution over binary event space but the probability estimation considers non-binary frequencies. The frequencies would naturally arise if we assume the document is a random sample from a multinomial distribution, and later language models were developed using multinomial models.

In a multinomial language model, the probability of query likelihood is computed as

$$p(q|\theta_d) = \prod_{i=1}^{n} p(q_i|\theta_d) \tag{2.9}$$

where q_i is i^{th} query term.

We can decompose the estimation of the probability of query generating from the document language model into two parts: (a) query terms that are present in the document and (b) terms that are not present in the document. By applying logarithm to the Eq. (2.9), we can rewrite the above equation as,

$$\begin{aligned} log\, p(q|\theta_d) &= \sum_{i=1}^{n} log\, p(q_i|\theta_d) \\ &= \sum_{i:c(q_i,d)>0} log \frac{p_s(q_i|\theta_d)}{p_u(q_i|\theta_d)} + \sum_{i=1}^{n} p_u(q_i|\theta_d) \end{aligned}$$

where p_s, p_u represents the probability model for the query terms present (seen) and not present (unseen) in the document respectively. The probability of unseen words computed using collection statistics, i.e., $p_u(q_i|\theta_d) = \beta_d\, p(q_i|C)$ where β_d is a constant and C is collection. These models are often referred to as smoothed language models.

The primary goal of the query likelihood language model is to estimate the $p(t_i|\theta_d)$, in an unsmoothed model the probability estimated as the relative frequency as described in Eq. (2.8). But, in practice a non-zero probability is assigned to unseen query words in the document using smoothing techniques. There are a number of smoothing functions that can be used to assign non-zero probability using collection frequency. Some popular smoothing techniques used in language models are the Jelinek-Mercer (JM) method and Bayesian smoothing using Dirichlet priors. JM smoothing involves a linear combination of maximum likelihood model with collection frequency model using a controlling variable as shown below,

$$p_\lambda = (1-\lambda)\, p_{ml}(t_k|\theta_d) + \lambda\, p(t_k|C) \tag{2.10}$$

where $\lambda \in [0,1]$.

Similarly, the language model with Bayesian smoothing using Dirichlet priors is given as,

$$p_\mu = \frac{c(t_k,d) + \mu p(t_k|C)}{\sum_{t:t\neq t_k} c(t,d) + \mu} \tag{2.11}$$

where Dirichlet parameters are chosen as $((\mu p(t_1|C)), \mu p(t_2|C), \mu p(t_3|C), \cdots \mu p(t_n|C))$.

Based on the above language models, a number of other competitive models were developed (e.g., [LZ01]). But the fundamentals of estimating the relevance between document-query pair is using common vocabulary space. And, these simple language models cannot use any relevance information - partly because of having no explicit notion of relevance [Rob03].

2.1.5 Lafferty/Zhai Equivalence of Language Models and RSJ

The formal lack of *relevance* in language models leads to the claim that the language models can be implicitly formulated using *relevance* variable [LZ03], and can provide an explanation of equivalence between RSJ and language models. In order to provide an explanation, the authors start by defining three random variables d, q, R for document, query and relevance respectively. And, both language models and RSJ models approximate the posterior distribution over relevance variable R and the factorisation of dependencies done differently.

According to the explanations, in the RSJ model, the factorisation is done as follows (as described in earlier sections):

$$\begin{aligned}\frac{P(R=1|d,q)}{P(R=1|d,q)} &= \frac{P(d|R=1,q)}{P(d|R=1,q)}\frac{P(R=1,q)}{P(R=0,q)} \\ &\propto \frac{P(d|R=1,q)}{P(d|R=1,q)}\end{aligned}$$

Where as in language models the factorisation is given by,

$$\frac{P(R=1|d,q)}{P(R=1|d,q)} = \frac{P(q|R=1,d)}{P(q|R=1,d)}\frac{P(R=1,d)}{P(R=0,d)}$$

To get the language modelling framework, the authors make the following two assumptions,

(a) query q is independent of d when $R=0$ (b) document d is independent of R

These two assumptions leave the above equation as,

$$\frac{P(R=1|d,q)}{P(R=1|d,q)} \propto P(q|R=1,d) \tag{2.12}$$

The equation Eq. (2.12) represents the probability that query q is generated from the document d, and represents query likelihood language models. In line with the explanation, the assumption of independence between the relevance and the document makes it difficult to use relevance information on both document and query simultaneously in language models [Lav04].

2.1.6 Generative Theory of Relevance (GTR)

The generative theory of relevance was proposed by Lavrenko [Lav04] in order to develop a generalised IR model. The principle behind the model is that the documents and queries are generated from a single common latent space. Therefore, the documents and queries are represented over the common feature space, e.g., vocabulary terms in the case of ad-hoc retrieval. The model estimates the probability of relevance between a document-query pair as follows:

(a) estimate the relevance model for the query, R_q (using the query terms). The relevance model is a distribution over the vocabulary terms in a simple ad-hoc retrieval scenario.

(b) compute the probability of relevance as the probability of sampling the document from the relevance model of the query, R_q.

The most important observation is that the representation of information need and item is over a *common* property space (vocabulary terms). Also, it does not provide a mechanism to modify the document representation based on relevance information over the document in order to estimate relevance of the document for future queries. Note that this model can use relevance information on the query (unlike the previous language models).

2.1.7 Logical Probabilistic Inference Model

The fundamental principle behind the logical view of an information retrieval system is that the retrieval of documents is interpreted as logical implications [vR86]. The retrieval function involves finding the documents that imply the query, i.e., $d \rightarrow q$ is true. Computing the exact implication is not possible due to the uncertain nature of the information retrieval problem, so, the model computes the probability of implication in retrieving the documents using $P(d \rightarrow q)$. The probability $P(d \rightarrow q)$ is computed as the conditional probability $p(q|d)$ [vR86]. In some cases, the logical models consider implication $P(q \rightarrow d)$ for retrieval and the probability of this implication is computed as $p(d|q)$. In this model, the probability $P(d \rightarrow q)$ and $P(q \rightarrow d)$ can be viewed as a measure of exhaustivity and specificity of the document respectively.

In order to do the inference in logical models, Wang and Yao [WY91] proposed a probabilistic inference model for IR. The basic idea is that it is assumed that there exists an ideal common concept space U, called the universe of discourse. Any given proposition is a subset of U. In a simpler case, each vocabulary term can be considered as a concept in U and a document d and a query q are two propositions containing a subset of concepts from U, i.e., $d, q \in U$. Under this model, a probability function P is defined over the concept space U. The probability $p(d)$ is the degree at which d represents the knowledge of concept space U. Similarly, $P(d \cap q)$ is interpreted as the degree to which the $P(d \cap q)$ encodes the knowledge of U. The logical models evaluate the relevance of documents using these probability measures over various set operations, i.e., the probabilistic measure of uncertain implication is considered as a measure of relevance in retrieving the documents.

In order to retrieve the relevant documents for a given query under the logical inference model, we can write the probability of relevance in three different forms as follows:

$$P(d \rightarrow q) = P(q|d) = \frac{P(q \cap d)}{P(d)} \tag{2.13}$$

$$P(q \rightarrow d) = P(d|q) = \frac{P(q \cap d)}{P(q)} \tag{2.14}$$

$$P(q \leftrightarrow d) = \frac{P(q \cap d)}{P(q \cup d)} \tag{2.15}$$

It is important to note that both $P(d \rightarrow q)$ and $P(q \rightarrow d)$ are asymmetric ranking function scores as $P(d \rightarrow q) \neq P(q \rightarrow d)$ whereas $P(d \leftrightarrow q)$ is a symmetric ranking function. The interesting relationship for us is the mutual support between d, q given by $P(q \leftrightarrow d)$. It provides a *symmetric bi-directional* implication based interpretation of relevance measure under the logical inference. The ranking function derived in the following chapters under our new theory of bi-directional matching is a symmetric relevance ranking function, capable of solving the problem of information matching.

There are two main observations in the logical inference model in relation to solving the problem of information matching: (a) the model computes the probabilities by representing both the documents and the queries in a common knowledge space (b) there is no straightforward way to simultaneously incorporate relevance information specific to a given document and a query in computing the probability of relevance. So, it does not satisfy the requirement of using relevance information on both query and document simultaneously as required for the unified relevance model.

2.1.8 Bayesian Inference Network Models

Bayesian inference networks have been described as the probabilistic formalism for inference networks with uncertainty [Pea88]. A document retrieval model based on an inference network has been developed by Turtle & Croft [TC90]. The fundamental concept of the model includes a network for each document and query. The document network represents the documents in the collection whereas the query network contains a single information need node connected to more than one query representing the same need.

Each document in the document network is represented by a document node and a set of text representation nodes and document concept nodes (generally, each concept corresponding to one or more text terms). Each node will take a boolean value either “true” or “false”. The value of a node depends on its parent nodes and an internal node-specific function to compute the value using its parent node values. The directed arcs of the network represent the probabilistic dependency of the network. Note that one document can have more than one text node as child nodes and each document concept node can have more than one parent text node (both text nodes representing the same concept). A query network will have a single leaf node representing the information need and multiple root nodes corresponding to the query concepts that express the same information need.

When doing inference in the network, the query network is attached to the document network with direct arcs from document concepts in the document network to query concepts in the query network.

Now, in order to estimate the probability of relevance between a document d and query q, the probability of $P(d \to q)$ is computed by setting the document node value as “true” and then the probabilities of depending nodes are computed until the probability $P(q = true)$. This model provides the fundamental concepts for using inference networks for document retrieval. Similar to other models, this model does not provide an explicit way to combine multiple sources of information to solve the problem of information matching or the unified model. However, in later chapters, we model our probabilistic bi-directional relevance model as a Bayesian inference network and provide an inference using Variational Bayesian methods.

2.1.9 Probabilistic Indexing

In information retrieval, probabilistic document indexing is the task of assigning weighted terms to documents for retrieval and these weights of indexing terms may be based on theoretical justification that is probabilistic in nature. There have been two influential probabilistic indexing models for information retrieval: (a) The Maron & Kuhns Model and (b) Probabilistic Learning model for document indexing

The Maron & Kuhns Model

A retrieval model based on a probabilistic indexing technique was proposed by Maron & Kuhns [MK60a] to index and search literature in a library system. The basic idea is that the probabilistic indexing weights for a document can be estimated on the basis of relevance information from a number of queries with relation to the specific document (relevance feedback on the document).

The model computes the weight for each term t_k in the document d with the probability that the user issued query q with term t_k will judge it as relevant. That is, the probability indexing weight of term t_k for given document d, $w(t_k, d)$, is computed as

$$w(t_k, d) = \frac{count(R = 1, t_k, d)}{count(\mathbf{Q}, t_k)} \tag{2.16}$$

where $count(R = 1, t_k, d)$ is the number of times d is judged as relevant for the queries issued by the users with term t_k and $count(\mathbf{Q}, t_k)$ is the number of queries with term t_k in the collection of all previous queries $\mathbf{Q}$ issued to the system. The term probability indexing weight $w(t_k, d)$ is computed based on the relevance feedback on document d, and is the same as the probability $P(t_k|\mathbf{Q}, R = 1, d)$.

For retrieval, the term indexing weight of query q terms for a given document d are used in computing the probability of the relevance between d and q in the following way,

$$P(R = 1|d, \mathbf{Q}, q) = \prod_{t_k \in q} \frac{P(d, R = 1|\mathbf{Q})P(t_k|\mathbf{Q}, d, R = 1)}{P(t_k|\mathbf{Q})} \tag{2.17}$$

$$\propto \prod_{t_k \in q} \underbrace{P(d, R = 1|\mathbf{Q})}_{\text{Prior}} P(t_k|\mathbf{Q}, d, R = 1) \tag{2.18}$$

$$\propto \prod_{t_k \in q} \underbrace{P(d, R = 1|\mathbf{Q})}_{\text{Prior}} w(t_k, d) \tag{2.19}$$

$$\tag{2.20}$$

The two parts on the right side of Eq.(2.17) are estimated using user feedback statistics computed using a relevant set of query-document pairs $\mathbf{R}$. The most important observation is that the model uses the *relevance* information on individual documents in estimating the probability of relevance between the document and the query. And, the formulation does not incorporate the relevance information (relevance feedback) on an individual query which is essential for solving our problem of information matching.

Probabilistic Learning for Document Indexing

A probabilistic learning method for document indexing was proposed by Fuhr [FB91]. The fundamental concept of the model is the abstraction of term-document relationships called *relevance description*. The advantage of this abstraction is that there could be more than one term-document pair with similar relevance description. This provides the model an easy way to estimate the parameters of term-document pairs as we can generalise the relevance description over a collection of pairs [FB91]. The parameter estimation problem is evident in Maron & Kuhns model where the indexing weight is estimated using the relevance information on only the single document.

In this model, the relevance descriptions contain values of features corresponding to documents, queries and terms. For example, a relevance description $x(t_k, d_m)$ contains the relationship between the term t_k and the document d_m. The model does not make any specific assumptions about the structure of x. Therefore, a description can be adapted in relation to the specific application, e.g., x can be the term frequency of t_k in d_m or x can be the inverse document frequency of t_k. Note that for index weighting, we are computing $P(R = 1|x(t_k, d_m))$ instead of $P(R = 1|t_k, d_m)$ where $P(R = 1|x(t_k, d_m))$ is the probability that a given document is judged relevant for a given query. By computing $P(R = 1|x(t_k, d_m))$, the model is considering the same relevance description x information from multiple document-query pairs. In general, the relevance descriptions $x(t_k, d_m)$ are formed using query-document pairs. And, the $P(R = 1|x(t_k, d_m))$ is computed by learning a classification function from the set of known relevant query-document pairs and their relevance descriptions.

The primary advantages of the model are that different documents or terms can be mapped to the same relevance description. This will avoid the problem of not having enough relevance judgements on a particular document or query. Another important advantage of the model is the adaptability of the relevance description representation. In traditional models, there is a specific form of representation of documents or queries and a different model has to be developed for every new piece of information to be incorporated into the model. This model has led to more successful retrieval models described as learning-to-rank models, such as Lambda-Rank [BRL06]. The probabilistic learning model for document indexing is also categorised as a description-oriented approach as it is based on the representation of descriptions of IR objects (documents, queries, terms), in contrast to the previously described model-oriented approaches.

In model-oriented approaches, the probability estimates are computed in relation to the representations whereas in description-oriented approaches a query-document pair is mapped to a feature vector of relevance descriptions. The problem of specifying the feature vector makes the model more heuristic compared to model-oriented approaches [Fuh92]. Furthermore, modifying the representation of an individual document or query based on the relevance feedback is not straightforward as the model maps them into a set of relevance descriptions.

2.1.10 Relevance feedback

The traditional notion of relevance feedback, which has been around since the work of Rocchio [Roc71] and was the original motivation for Model 2, considers one user-need only. The user is offered some items resulting from an initial search and invited to provide relevance judgements; these are used to improve the formulation of the search query (typically terms and weights), which is then re-run to find new items to offer the user. This notion has been investigated in many experiments, for example in the SMART laboratory where Rocchio was based, in the Okapi work [Rob97a], and in the TREC routing and adaptive filtering tracks. In the early years of TREC, a spin-off notion called Pseudo Relevance Feedback (where the top-ranked documents are assumed to be relevant without human judgements) developed into one of the main components of much further work on improving search.

The second prior model [RJ76], Model 2, is expressed from the point of view of an individual user making a request. If we have any examples of items that this user would like to see, they can be used to characterise the kinds of items that the user wants. New documents should then be selected and ranked based on the probability that they will satisfy the user's information need.

In modern web search, by contrast, there is little use of this form of relevance feedback (either real or pseudo). On the other hand there is very substantial use of a form of feedback that relates to Model 1. This does not ask for explicit human relevance judgements, but nevertheless infers something related to such judgement from user actions. A user posing a query and clicking on a result is taken as (maybe weak) evidence that this result is relevant or appropriate for that *query*. The emphasis here is because the evidence provided by the click will probably not be used in this particular session (particular instance of a user information need), but will instead be associated with this particular textual query, and will (taken cumulatively) affect the results seen by a later user issuing the same query. Traditional probabilistic relevance models do not incorporate this feedback explicitly. When we combine the two models, we have a matrix of user needs (rows) against items (columns), with some notion of groups of similar user needs or similar items as shown in fig. 1.1 and a unified model should be able to consider this information while computing the probability of relevance.

2.1.11 Unified Models

Fuhr [Fuh86] takes on the unified model notion of a matrix of individual user-needs against individual documents, but emphasises the need to learn in a generalised way that can be applied to new items and new user requests, and therefore the need to abstract away from individuals in both cases. His methods are based on 'descriptions' of items and user needs, which in principle can apply to multiple individuals, and somewhat generalise the notion of features. The underlying principle behind the representation of information objects in our model strongly correlates to the ideas of user need and item information representation in his model.

Bodoff and Robertson [BR04] start from existing features of both documents and user needs, and modify them in a gradual way. Each feedback or learning instance (relevance judgement) is the subject of a credit-assignment decision, to apportion its effect between modification of the document and modification of the query. Robertson [Rob04] looks at the notion of probability of relevance and raises the issue of the event space(s) in which such a notion can be defined. Luk [Luk08], Boscarino and de Vries [BV09] and Aly and Demeester [AD11] further explore the issue. All the above attempted unified models have produced interesting modelling ideas but none succeeds completely [Rob03].

2.2 Collaborative Filtering

Recommender (collaborative filtering) systems have some similarities to, as well as some significant differences from, information retrieval systems. In recent years, Collaborative Filtering (CF) algorithms have become the hallmark of web-based recommender systems. These techniques compute personalised recommendations for users by learning from the ratings or interactions that the users create as they engage with the (movie, music, news) content on the system [GNOT92, SK09]. In both types of systems, IR and recommendation, we need to satisfy the requirements of a particular user by offering him/her particular items from a collection.

Collaborative filtering is based on the assumption that, as users give ratings for items on a web site, their discovery of new items can be aided by learning from the ratings that other users have given. Broadly speaking, CF uses a matrix of user-item ratings in order to compute predictions for those items that users have not rated; items can then be ranked, in descending order, by these predictions. We can look at this as a problem of probabilistic relevance ranking of items for a given user where the only available information for estimating the probability is the *relevance feedback* on users and items in the form of ratings.

In the following sections, we describe two state-of-the-art collaborative filtering approaches/algorithms that pervade the literature in the field of personalised recommendation: neighbourhood approaches and latent factor models. The neighbourhood approaches are based on the similarity between users or items. The latent factor model, instead, represents the user and item in the same latent space with a predefined number of hidden dimensions. This is generally a lower dimensional space and the rating between the user-item pair is predicted by the proximity of the relevant latent factors.

2.2.1 Neighbourhood Models

Before we describe the models, we define the following notation which we use subsequently in this section. Let $\mathcal{U}$ and $\mathcal{V}$ be the collection of users and items/products respectively. By u_m and v_n we denote m^{th} user in $\mathcal{U}$ and n^{th} item in $\mathcal{V}$ respectively where $m \in [1, |\mathcal{U}|]$ and $n \in [1, |\mathcal{V}|]$.

The k-Nearest Neighbourhood algorithm has been used in recommender systems research since its inception [DK04]. There are two different flavours of the algorithm: *user* and *item*-based. The user-based approach represents users as a sparse, high-dimensional vector of item ratings. The item-based alternative, instead, represents items as sparse, high dimensional vectors of user ratings. Both methods operate by measuring any similarity between the users or items in order to predict a score for not rated items. The similarity measure often used is either Pearson correlation or cosine angle between two vectors.

Specifically, for example, the cosine similarity between users u_m and u_s is measured as:

$$sim(u_m, u_s) = \frac{\Sigma_{v \in I_m^s}(r_{mv} - \hat{r_m})(\hat{r_{sv}} - \hat{r_s})}{\sqrt{\Sigma_{v \in I_m^s}(\hat{r}_{mv} - \hat{r_m})^2}\sqrt{\Sigma_{v \in I_m^s}(r_{sv} - \hat{r_s})^2} \tag{2.21}$$

where I_m^s is the set of items rated by both users u_m, u_s and $\hat{r_m}, \hat{r_s}$ are the average rating of co-rated items of u_m and u_s respectively.

Once similarities have been computed, the predicted rating for a user-item pair is computed as the similarity-weighted average of the ratings that (in the item-based approach) the user has given to the k most similar items or (in the user-based approach) the ratings that the k most similar users have given to the item. More formally, the predicted rating $\hat{r}_{u_m,v_n}$ for user u_m and item v_n, in the user-based approach, is computed with:

$$\hat{r}_{mn} = \bar{r}_m + \frac{\Sigma_{j=1}^{N}\left(sim(u_j, u_m)(r_{jn} - \bar{r_j})\right)}{\Sigma_{j=1}^{N} sim(u_j, u_m)} \tag{2.22}$$

where $\bar{r}_m$ is the mean rating of user u_m, and u_j is the j^{th} neighbour of u_m. The neighbourhood size N is usually set to 50 for both approaches whenever we do experiments.

2.2.2 Latent Factor Models

Many recent recommender algorithms have been based on latent factor models [KBV09]. The fundamental assumption behind these models is that users and items are represented in a common latent factor space. The idea behind these models is to factorise the rating matrix into two lower rank matrices, one capturing user factors ($\mathbf{p}_{u_m}$), and one for item factors ($\mathbf{q}_{v_n}$). Each user and item is represented over a fixed $f-$dimensional feature space, where f is a parameter to the model, set to keep the dimensionality low. For example, the common practice is to set $f = 20$ in collaborative filtering. A predicted rating for a user-item pair (u_m, v_n) is then computed with the inner product between the related factor vectors:

$$\hat{r}(u_m, v_n) = \mathbf{p}_{u_m}\mathbf{q}_{v_n}^T \tag{2.23}$$

The majority of recent models learn the factor vectors using any available ratings and an objective function. To illustrate the model, later, we follow Koren *et al.* [KBV09] and learn the vectors' factors via a gradient descent objective function. To be consistent with the literature, we will refer to this model as *PureSVD*.

2.2.3 Content-Based Recommendations

The content-based recommender systems recommend items to the user based on the similar content description of the items that the user has preferred in the past [PB07]. For example, a user can be recommended a movie with the same genre as the movie he has preferred before. These methods operate on domain specific knowledge such as genre and location in order to infer the relationship between the items for recommendation. In general, it is often the case that the content description of an item is combined with rating information in computing the recommendation. These techniques are often referred to as hybrid recommender systems. The prominent way of combining content with other information in recommendation is by learning the two class classifier of like/dislike on user-item pairs using the user-item pairs content description features and classifying unseen pairs based on their content features [BHC98]. The theory and probabilistic model proposed in this thesis will naturally combine multiple sources of information and the same model can be applied to both recommendation and text retrieval or in general any information matching problems.

2.2.4 The Popularity Model

A simple non-personalised model used for recommendation is the *popularity based model*. The idea behind the model is simple: rank the items based on their popularity and present the same set of popular items to each user. The popularity of an item is defined as being proportional to the number of ratings that it has received by any user; by counting ratings, we can rank items in descending order of popularity. In the following, we will refer to this baseline as *Pop-item*.

2.2.5 Group recommendation

While traditional research on recommender systems has focused on finding relevant items for *single* users, there is an increasing occurrence of individual web profiles and accounts being shared amongst a group of people. For example, a household of users may share a single movie-rental and recommendation account, or users of mobile recommender systems may be seeking locations (e.g., restaurants) for a group of friends to go to together. Both of these types of scenario have led to recent research on *group recommendation* [JS07]. Group recommendation scenarios tend to differ in terms of how preference data about the group is collected. For example, in the case of shared household web accounts, the ratings that the system represents as a single user may actually reflect a number of people's preferences; in the case of a restaurant-recommender system, multiple profiles may need to be considered simultaneously in order to suitably personalise the system's recommendations. Group recommendation systems have been used in various forms across the web to recommend news pages [PDCCA05], holidays [MSC$^+$06], music [CBH02], and both TV programs and movies [Sai11, OCKR01, YZHG06].

The main challenge behind these scenarios has been that of computing the relevance of items to the group from a potentially diverse set of group members. Here, the probabilistic relevance model should compute the relevance of the items to each individual member of the group and compare the scores of items to the individual members. This requires a probability space in which the relevance scores of items to the group members are comparable. This is not possible to achieve with the traditional relevance ranking models as the relevance estimation is always conditioned on either a single user or single item but not a group of users, and requires a *unified modelling framework*. Later we will explore this aspect in detail.

Past techniques have tackled the problem of group recommendation in one of two ways. On the one hand, all group members' ratings can be folded into a single profile (Merging Profiles), which can then be treated as a unique user that recommendations should be computed for. Naturally, this approach may bias its output away from those group members that have the sparsest profiles. Alternatively, personalised recommendations can be computed for each group member first, and the resulting set of ranked recommendations can be merged into a single list for the group using pre-defined heuristics (merging recommendations) [BMR10, JS07]. This solution assumes that successful group recommendation comes from the strategies used to merge the ranked lists, without modelling the group as a whole or considering how individuals' preferences may differ when they become part of a group. The following explains both the strategies with simple examples.

Merging Profiles: If we have two users, who have respectively rated $\{i_1, i_2\}$ and $\{i_3, i_4\}$, then their group profile is simply $\{i_1, i_2, i_3, i_4\}$. If, instead, there is an overlap in the group member's ratings, such as if they have rated $\{i_1, i_2\}$ and $\{i_1, i_3\}$, then the merged profile is $\{mean(i_1), i_2, i_3\}$. The resulting profile can then be directly used in any of the CF algorithms as if it were a single user.

Merging Recommendations: A variety of aggregation techniques have been proposed in the literature [Mas04, DKNS01, JS07, BMR10]. Later, we use the strategy known as *Least Misery* (LM). This approach seeks to minimise the probability that any one member of the group will strongly dislike the recommendations. More formally, once we have generated a set of recommendation lists for members of a group G, we set the relevance score of any item i as the smallest relevance score from amongst the group's individual's relevance scores. This means that the relevance of an item to a group is the least satisfied member's score.

Recent work has compared the two strategies: in [BF10], the authors found that the first approach marginally outperforms the second when using recipe ratings collected from a variety of families. The work in [BMR10], instead, concludes that the LM strategy outperforms a range of other techniques.

All of the above group recommendation approaches tend to assume that peoples' preferences do not vary depending on whether they are alone or in a group. We will revisit this assumption by incorporating group membership into a probabilistic model. The authors in [AYRC$^+$09] also addressed this aspect of groups by explicitly integrating a *group disagreement* score into the groups' relevance rating score. Note that the group information is contextual and there is no straight forward way to incorporate it into latent factor models or neighbourhood models used for recommendation.

There are number of comprehensive studies to analyse these models and provide the relationship between different models in information retrieval. For more examples, two recent papers by Roelleke [Röl13] and Li [LX14] provide good insights and complement each other in giving an overview of theoretical aspects of these models and their application in web search engines as part of machine learning algorithms.

2.3 Discussion

From the previous discussion, a variety of solutions has been developed to estimate the relevance in one way or the other based on a part of the total information that can be used. For example, neither in text retrieval [RSJ76, PC98, LZ01] nor in recommender systems [KBV09] are state-of-the-art models designed to accommodate all the information that may be available in estimating the relevance. The traditional probabilistic retrieval relevance models cannot accommodate different kinds of information for estimating the relevance due to the fact that they are designed to use information over common attributes (e.g., vocabulary terms).

In the case of information retrieval, we usually start from features (often words), but may also make use of user feedback (relevance feedback). In the case of recommendation, we usually start from feedback (user ratings) but may also make use of features. The most common approach to the task of recommendation relates strongly to information retrieval [WdVR08]. Given that in many recommendation situations we lack features that could be used directly, it is common to attempt to derive a set of hidden features which might explain the ratings that we observe, and use them to predict new ratings, from either a probabilistic [Hof04] or non-probabilistic perspective [KBV09]. These features are usually assumed to describe both users and items, so that both entities may be embedded in the same space – this parallels the information retrieval situation, where users (in this case user queries) and items both have words as features, and we consider both entities as points in a space defined by words. The usual assumption in such recommendation systems is that this space is of relatively low dimensionality; although this assumption is by no means universal in information retrieval, it is well represented there in the form of topic models such as PLSI [Hof99] and LDA [BNJ03]. One of the fundamental issues with these approaches is that the modelling of the user or item with interpretable features is not straightforward. For example, if we want to use a user's *age* as a feature, the only way is to model the age as an independent user, and represent it in a low dimensional space [CZL+11], which is not a natural way of using the information.

The fundamental problem all these models are trying to address is to find the relevance match between two distinct information objects, e.g., user and product. The above discussed models are designed to solve this problem of estimating the relevance between two objects by matching their properties, but each model is subject to the type of information available to the system. For example, BM25, Language models and Maron & Kuhns model for probabilistic indexing are developed to solve the ad-hoc text retrieval problem [RZ09, MK60b, PC98]. All these ad-hoc retrieval models are designed to operate by representing documents and queries over a common vocabulary space, and use relevance information either on the document or on the query but not both [Rob03]. Similarly, Collaborative Filtering approaches are designed to perform the relevance matching between two information objects (e.g., user, movie) when there are no common attributes. Additionally, for the task of group recommendation, any relevance matching model should combine group relevance information with individual group members' preferences while computing the relevance of an item to the group.

A quick observation from the above discussion is that we don't have a formal probabilistic unified relevance retrieval model able to incorporate all the available information on information objects in computing the probability of relevance. The essence of the idea of a unified model is to combine all the available information in a way applicable to both text retrieval and recommendation tasks.

In conclusion, the underlying structure of the problem of estimating the probability of relevance between a user-product and query-document in the e-commerce and text retrieval scenario is identical, but the available information is different and not always complete. There are a number of problems with similar structure: for example, matching people for dating, matching jobs to job seekers and matching ads with users. In all these cases we have information about two entities/objects of different types (e.g., user and an ad) and their interactions (e.g., which user clicked on what ad). We need to estimate the probability of relevance (match) using all the available information. A solution to this problem would be significant and applicable in more than one domain. We regard this as a general problem of *information matching* for matching two information objects. In the next chapter, we formally define the problem of information matching by generalising all the matching problems previously described, and we provide a solution. As stated, the primary goal of this thesis is to develop a theory and a formal probabilistic relevance framework to solve the problem of information matching. A solution to the problem is a unified relevance retrieval model [RMC82a].

Chapter 3

Theory of Information Matching

In this chapter, we formally define the problem of *information matching* as described in Chapter 2 and present a solution by proposing a new theory of information retrieval called “Theory of Information Matching” (TIM). We start the chapter by providing a formal definition of the problem, and then explain various components of the new theory through some simple examples. We then derive a probabilistic unified relevance retrieval model from it. Finally, we conclude the chapter with some discussion.

3.1 Problem of Information Matching

Let $\mathcal{A}$ and $\mathcal{B}$ be the sets of two different types of information objects, and let $\mathcal{O}_a$ and $\mathcal{O}_b$ be two information objects, where $\mathcal{O}_a \in \mathcal{A}$ and $\mathcal{O}_b \in \mathcal{B}$. For example, $\mathcal{A}$ and $\mathcal{B}$ are a set of queries and documents or users and products respectively.

Problem 3.1.1 *Estimate the probability of relevance between $\mathcal{O}_a$ and $\mathcal{O}_b$ given the information about: (a) $\mathcal{O}_a$ (e.g., terms of a given query), (b) $\mathcal{O}_b$ (e.g., terms of a given document), and (c) all the interactions between the objects of $\mathcal{A}$ and $\mathcal{B}$ ($\mathcal{A} \times \mathcal{B}$) (e.g., relevance judgements on query-document pairs).*

The objective of the problem is to unify all the information for estimating the probability of relevance (degree of relevance matching) between two information objects, $\mathcal{O}_a$ and $\mathcal{O}_b$. In addition, a solution to the problem must be able use any partially available information about the objects. In the context of the problem, the *relevance* is interpreted as how well a retrieved information object satisfies the information need of another object as described in Chapter 1.

3.2 An Introduction to the Theory

In problem 3.1.1, in general, we see two different populations ($\mathcal{A}$ and $\mathcal{B}$) which need to be matched, together with some observable features of each (potentially from different populations or “vocabularies” of features). To make this introduction simple, we will borrow terminology from the recommendation example and refer to these two populations as “users” and “items”. This is just a matter of convenience, and we expect to apply the same model to the IR task (user information needs/items or documents or webpages), to the job-seekers task (job-seekers/employers with vacancies), to dating services (perhaps women/men). As we shall see in a moment, we assume a strong symmetry between the two populations, so that in any of these cases, the two could be reversed without changing the basic model. We note that the task could involve matching elements of the same population rather than two different ones; if the matching is pairwise, then the approach should still apply.[1] We first look at the job-seeker/vacancy example, because it exhibits a kind of symmetry not apparent in some of the other tasks, but which will be useful in the subsequent discussion.

3.2.1 Symmetry

Consider job-seekers looking for employment and employers looking for people to fill vacancies. Each job-seeker will have features (such as education, experience, abilities, constraints); and each vacancy will have features (such as job description, salary, location, working environment). Note that the vocabulary of features applicable to job-seekers is rather different from the vocabulary of features that might describe vacancies. However, each job-seeker also has desires and expectations regarding the features of the vacancy s/he would like to find; and similarly, each employer has requirements on the features of any potential employee in this vacancy. Temporally, a search can happen either way round: a job-seeker might search a database of vacancies, or an employer might search a database of job-seekers. The logical symmetry between job-seekers and vacancies is almost perfect – and extends to one further aspect. When it comes to judging the success or otherwise of a match, both parties must agree – ultimately, success can only be claimed if both employer and job-seeker agree that the match is good.

The dating agency task has a similar symmetry. In general, a person looking for a soul mate has preferences (certain qualities/features) for their potential partner. In order to be a perfect/successful match, it is indeed important that the preferences must satisfy both parties. The bi-directional preferences have to match in order to be a successful outcome [2]. In more general terms, a successful outcome can be two persons going on a date or a user buying a product in a retail scenario or the employer hiring the job-seeker for a position.

This symmetry of evaluation appears at first glance not to apply in many other contexts. Thus in traditional retrieval, the user/requester is normally taken to be the sole judge of success – neither the document nor the document's author is taken to have any say in this question. However, it can be argued that authors do indeed have some expectations about their readers. Occasionally, these expectations are given explicit form[3] , but they may well be implicit in the way the author writes. For example, in an educational context, authors normally write for readers who have reached particular academic levels or with particular prior knowledge. Even the language of the document may be seen in this light: I write for people who are sufficiently versed in English to understand my usage of this language, and those who are not will not be able to make use of my document, however relevant it might be in a topical sense. In

[1]An example occurs in the dating agency context. Typically dating agencies or lonely hearts columns operate four different matching tasks: women seeking men, men seeking women, men seeking men and women seeking women. In each of the latter two cases, the two populations involved are of course the same.

[2]Note: The personnel in question could be either women or men depending on their sexual orientation, and the above condition for a successful match holds for all!

[3]Some years ago, there was a book called *Pascal from Basic*, which aimed to help programmers learn the programming language Pascal, given that they were already familiar with the programming language Basic. 'Basic' was not a feature or characteristic of the book in the usual sense – it was not 'about' Basic, did not aim to teach its readers Basic. Rather 'Basic' specifies a feature of the intended readers of the book – a prerequisite feature without which the book would not be useful to them.

the case of serving ads from a webpage, the advertiser certainly has a stake in questions of success or failure.

In our theory, we assume that in the general case there are indeed requirements both ways round, and that success (even if eventually judged by just one party) has implicit components in both directions. Thus a user likes certain kinds of items, and also an item "likes" certain kinds of users. A successful outcome (a presented item is relevant to a user need, say), even if judged only by the user, actually depends on both the user's likes or preferences and the item's 'preferences' or propensities. From the above examples, finding the relevance between two information objects[4] is the same as the task of checking whether the bi-directional preferences of both objects are satisfied or not. In simple terms, in order to estimate the relevance between two information objects, our theory proposes to identify the preferences of information objects and verify whether they satisfy their bi-directional preferences.

The idea of bi-directional preference matching between two objects for relevance estimation seems natural but it poses a fundamental challenge of identifying/learning the preferences of both information objects. This challenge is apparent especially when two objects are represented in two distinct feature spaces, e.g., there are no common explicit features between users and items in a collaborative filtering (matching) problem. The problem is, how do we represent the preferences of both objects in order to facilitate a model to verify the bi-directional preferences for a relevance match? A solution to this problem can update the preferences of both objects (user and item) simultaneously when we observe relevance on either of two objects. This essentially means that the models based on bi-directional preference matching are capable of using the relevance information to update both information objects. We will illustrate this with examples after we present our theory. In the following section we describe how we solve this problem while describing the formal elements of our theory.

3.2.2 Formal elements, Preferences, and Relevance

We assume, as indicated above, that there are two populations of interest, described as "users" and "items". We assume that each object (user or item) has native features, from a feature set specific to that population. These are a-features and b-features respectively. For example, a given user can be described with a binary feature indicating whether she is a child or not. Similarly, an item can be represented with a binary b-feature like whether a given movie has *cartoon* as genre or not. The set of potential a-features includes all the items as well as other features; the set of b-features includes all the users. In general, feature values may or may not be observed – that is, there may be applicable features whose values have not yet been observed (most items considered as a-features may not yet be observed, and conversely). We would like to stress the fact that a-features and b-features can potentially be disjoint sets.

We assume two preference functions, a-preference and b-preference. Both a-preference and b-preference are actually functions on a-feature, b-feature pairs, with a range in some set of preferences, possibly binary (like/dislike). However, the relation of each to the elements of the pair differs. a-preference expresses the *kinds* of items that *individual* users like or dislike; the b-feature in the pair describes an example of the kind of item *liked* by a user with a-feature. From the above example, a-

[4]An information object is a representation of a person or product or a document or query

preference function will be like saying a user with a-feature *child* likes a movie with b-feature *cartoon*. And b-preference expresses the reverse, the *kinds* of users that *individual* items like. As per our assumption, all the information about user and item is expressed though a-features and b-features respectively. So, we represent a-preferences as a function of a-features to b-features that describe user and item respectively. Similarly, we define b-preference as a function of b-features to a-features. Essentially, we are representing a given user's preferences with a combination of user specific features (user representation over a-features) and their preferences to b-features through the a-preference function. This representation solves the problem of user and item not being represented in the common feature space and required to match their bi-directional preferences. This is possible as we are expressing the preferences as a function of two distinct feature sets. Next, we assume a relevance function from a user-item pair to a relevance value (again possibly binary) and define a logical relation between the preference functions and relevance, which will normally be a form of boolean **and**: that is, an item is relevant to a user only if *both* the user prefers that kind of item *and* the item prefers that kind of user, i.e., *bi-directional preferences* must be satisfied.

To illustrate the relevance as a function of preferences, consider an example of a salesman selling *luxury* products, say a Ferrari 150. A good salesman understands a targeted customer for his products. And, in this case, they are typically customers with a higher income. Now, consider a student who is very enthusiastic about sports cars (including Ferrari 150). However, for the sake of the example assume that he cannot afford to buy a Ferrari. Let us assume a scenario where the student walks into the car showroom and asks the salesman to recommend a car. In order to recommend a car, the salesman has to understand the interests and needs of the customer (student) and think about the cars that satisfy him. In this particular case, the salesman cannot (will not) recommend a Ferrari 150 because he will not be able to make a sale as the student cannot pay the price even though the student is very much interested in owning a Ferrari. From the example, we conclude that, in order to make a successful sale:

- The salesman has to look at the preferences of the student (what he is looking for and identify his need).
- Identify the cars those are made to satisfy the needs of student using the knowledge provided by the manufacturer (including the price).
- Find the best matching car by looking into the preferences of student as well as the car.

As we described earlier, it is similar to finding a best match for an individual looking for a relationship, i.e., the match making person has to consider the preferences of both the individuals and match based on the bi-directional preferences. From the above analogy, assume that our relevance matching system is a virtual salesman. And, it has to understand the preferences of both the student and the car, and find the match based on their bi-directional preferences by determining whether it is a successful match or not.

Why is it necessary to decompose the notion of relevance feature sets by means of these two preference functions? We have seen a clear and explicit requirement for such a decomposition in the case of the job-seeker task, and the same applies in the dating agency task. In fact in both these cases it would be possible to ask the people concerned for judgements of preference, quite separately from the judgement of ultimate relevance determined by a successful outcome. In the other tasks discussed it is less obvious. However, it sits well with the aim of using relevance judgements (in IR) or ratings (in recommendation) both ways round, to learn about both the users and the items. As Fuhr [FB91] has argued, we need some level of abstraction away from individuals. Moreover, we will show that this decomposition of relevance by means of two preference functions and the representation of user and item with the separate features sets allows the relevance models to use all the available information in computing the relevance between two objects – thus solving the problem of information matching in 3.1.1. In the following sections, we present a formal framework for our theory, and then we derive a probabilistic model using the framework. We also show how this probabilistic model would incorporate all the available information in computing the relevance between two objects.

3.3 A Unified Information Matching Framework

Before we formally present our model based on the ideas presented in the previous sections, we start by stating our hypothesis in the following way,

Definition 3.3.1 (Information Matching Hypothesis (IMH)) *: Any information need or information item can be described using two distinct sets of binary (elite) properties, called need and item properties respectively. The relevance between an information item-need pair is dependent only on the bi-directional preferences of the need and item properties that describe them.*

As we assume a strong symmetry in our model, we would like to name information need and information item as information objects or just objects. So, any reference to objects means either information need or item or a user or a product. If the two objects are from different populations then we explicitly specify.

From the hypothesis, essentially, we are stating that the need and the item are represented over two distinct sets of properties (or features or concepts). The block diagram in Figure 3.1 shows the representation of two objects and the preference mappings in the model. To be consistent with the 2-Poisson model [RW94] for now we name these binary properties as "elite" properties. The value of an elite property (referred as eliteness) for a given information object is binary and its value is "1" or "elite" if the property describes the object, "0" or "non-elite" otherwise. The second important observation from the IMH is that the *relevance* between two objects is dependent only on the 'elite' properties that describe need and item objects respectively (i.e., eliteness value is '1'). If we are able to deterministically identify the 'elite' properties of both the objects, then from the hypothesis we would be able to determine the relevance between these two objects by verifying their directional preferences.

3.3.1 Verifying bi-directional preferences for relevance

Let r_{ab} be the relevance between $\mathcal{O}_a$ and $\mathcal{O}_b$. We defined the relevance as a function of the preferences of 'elite' properties of $\mathcal{O}_a$ and $\mathcal{O}_b$. In order to verify the preferences, we define that $\mathcal{O}_b$ satisfies the preferences of $\mathcal{O}_a$ if and only if $\mathcal{O}_b$ is described by all the properties that are preferred by all the elite properties of $\mathcal{O}_a$. For example, if a user has only one elite property "child" and "child" prefers "cartoon" as a genre in a movie then any movie with genre (or described by) "cartoon" satisfies the preferences of the user. Similarly, $\mathcal{O}_a$ satisfies the preferences of $\mathcal{O}_b$ if and only if $\mathcal{O}_a$ is described by all the properties preferred by all the elite properties of $\mathcal{O}_b$. Under IMH, $r_{ab} = 1$ if and only if the preferences of $\mathcal{O}_a$ and $\mathcal{O}_b$ are satisfied, "0" otherwise. In the following section we mathematically express the framework and define a relevance matching function that can solve the problem expressed in 3.1.1.

3.4 Mathematical Formulation

We now formally develop a matching model, called the *information matching model* (IMM) or the *information matching framework*(IMF) model, based on IMH. As our model/framework is applicable to many different matching tasks, we would like to present it as a generic framework for matching two distinct types of information objects.[5]

Before describing the mathematical formulation, we define a consistent notation wherever it is necessary and use it subsequently. From the definition of the problem, we have two distinct types of objects represented by $\mathcal{O}_a$ and $\mathcal{O}_b$ where $\mathcal{O}_a \in \mathcal{A}$ and $\mathcal{O}_b \in \mathcal{B}$. From the hypothesis, we represent each object $\mathcal{O}_a$ over a set of a-features and $\mathcal{O}_b$ over a set of b-features. Now, let $\mathbf{E}, \mathbf{F}$ be the sets of a-features and b-features respectively where $\mathbf{E} = \{e_1, \cdots, e_K\}$ and $\mathbf{F} = \{f_1, \cdots, f_L\}$. From the definition: (a) e_k and f_l are individual features that describe corresponding information objects where $k \in [1, K]$ and $l \in [1, L]$. The values of these features for a given object are binary. (b) $\mathcal{O}_a \in \{0,1\}^{|\mathbf{E}|}$ and $\mathcal{O}_b \in \{0,1\}^{|\mathbf{F}|}$. Let $\mathbf{G}$, $\mathbf{H}$ be the a-preference and b-preferences functions respectively where $\mathbf{G}(k,l) \in \{0,1\}$ and $\mathbf{H}(l,k) \in \{0,1\}$. Note that $\mathbf{G}$ and $\mathbf{H}$ are a function of a-preference and b-preference.

We know that the value of e_k for a given object $\mathcal{O}_a$ is binary. Let this binary value for the k^{th} feature be denoted by α_k^a where

$$\alpha_k^a = \begin{cases} 1, & \text{if } e_k \text{ describes the object } \mathcal{O}_a \;\; (e_k \text{ is elite for } \mathcal{O}_a) \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Similarly, let the binary value of f_l for $\mathcal{O}_b$ be denoted by β_l^b where

$$\beta_l^b = \begin{cases} 1, & \text{if } f_l \text{ describes the object } \mathcal{O}_b \;\; (f_l \text{ is elite for } \mathcal{O}_b) \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

Finally, we know that $\mathbf{G}$ and $\mathbf{H}$ are the functions of e_k and f_l. Let γ_k^l and δ_l^k be the binary function value of $G(k,l)$ and $H(l,k)$ respectively. From our theory, we define the preference functions as follows,

$$\mathbf{G}(k,l) = \gamma_k^l = \begin{cases} 1, & \text{if } e_k \text{ prefers } f_l \\ 0, & \text{otherwise} \end{cases} \qquad \mathbf{H}(l,k) = \delta_l^k = \begin{cases} 1, & \text{if } f_l \text{ prefers } e_k \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

We can also view the preference functions as a matrix as follows, the binary a-preference matrix is defined as:

[5]We stress the fact that the model is applicable to objects of the same population without any change!

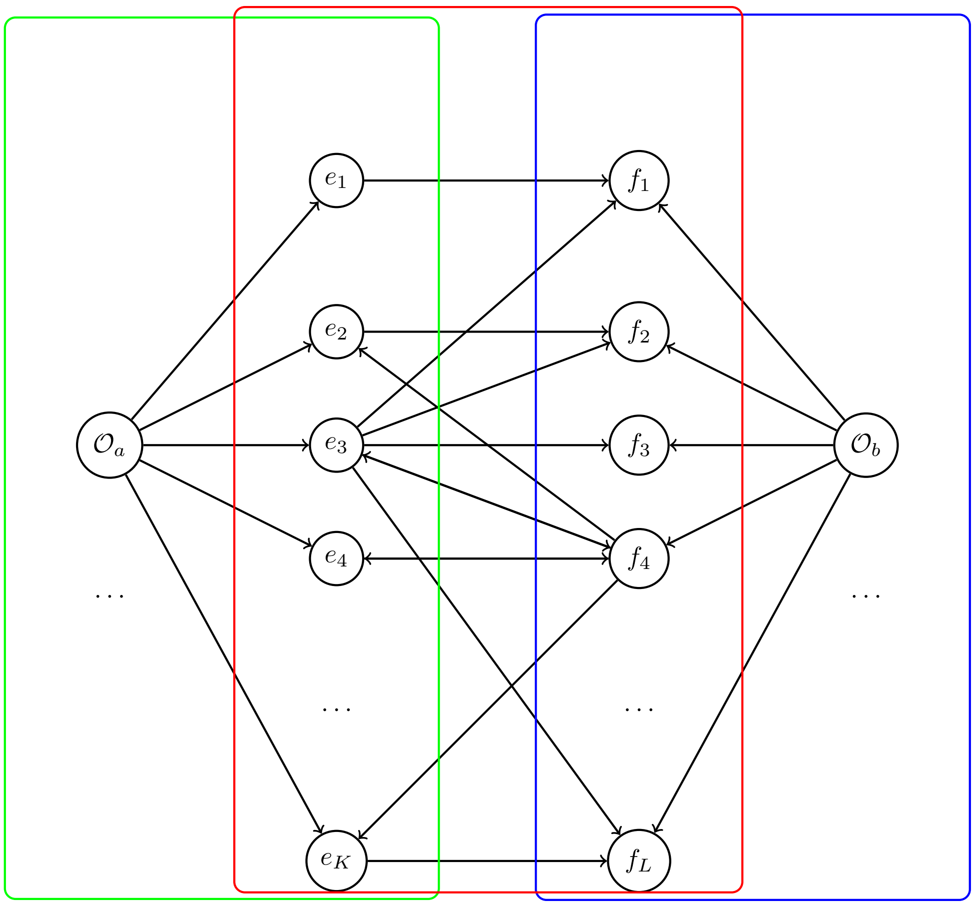


Figure 3.1: The block diagram depicting the model representation of two distinct types of objects with their corresponding properties and the preference mappings. The left box is a representation of an object of type one with K features ($\mathbf{E}$), the right box is a representation of an object of type two with L features ($\mathbf{F}$), and the middle box represents the preference mapping space between the two sets of properties.

$$\mathbf{G} = \begin{array}{c} \\ e_1 \\ \vdots \\ e_K \end{array} \begin{array}{c} \begin{array}{cccc} f_1 & f_2 & \cdots & f_L \end{array} \\ \begin{pmatrix} \gamma_1^1 & \gamma_1^2 & \cdots & \gamma_1^L \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_K^1 & \gamma_K^2 & \cdots & \gamma_K^L \end{pmatrix} \end{array}$$

Similarly, the binary b-preference matrix is defined as:

$$\mathbf{H} = \begin{array}{c} \\ f_1 \\ \vdots \\ f_L \end{array} \begin{array}{c} \begin{array}{cccc} e_1 & e_2 & \cdots & e_K \end{array} \\ \begin{pmatrix} \delta_1^1 & \delta_1^2 & \cdots & \delta_1^K \\ \vdots & \vdots & \ddots & \vdots \\ \delta_L^1 & \delta_L^2 & \cdots & \delta_L^K \end{pmatrix} \end{array}$$

Here, the relationships between these properties are global and not dependent on any particular object.

3.4.1 Relevance function

To determine the relevance between two objects using the above formulation, we check the directional preferences of an object as follows: $\mathcal{O}_b$ satisfies the *preferences* of $\mathcal{O}_a$ if and only if $\mathcal{O}_b$ is described with

	adult (e_1)	*child* (e_2)
John $(\mathcal{O}_a^1)$	1 (α_1^1)	0 (α_2^1)
Emily $(\mathcal{O}_a^2)$	0 (α_1^2)	1 (α_2^2)

Table 3.1: Users' features

	cartoon (f_1)	*horror* (f_2)
Alien $(\mathcal{O}_b^1)$	0 (β_1^1)	1 (β_2^1)
Cars $(\mathcal{O}_b^2)$	1 (β_1^2)	0 (β_2^2)

Table 3.2: Movies' features

An example set of users, movies and their features in our theory.

all the elite properties (eliteness is "1") preferred by the elite properties of $\mathcal{O}_a$. In simple terms, if α_k^a is elite for $\mathcal{O}_a$ ($\alpha_k^a = 1$) and $\mathbf{e}_k$ prefers f_l ($\gamma_k^l = 1$, i.e., $e_k \rightarrow f_l$) then f_l must describe $\mathcal{O}_b$ (f_l must be elite to $\mathcal{O}_b$). Similarly, $\mathcal{O}_a$ satisfies the preferences of $\mathcal{O}_b$ if $\mathcal{O}_a$ satisfies the preference conditions of all the elite properties of $\mathcal{O}_b$. We can represent both preference conditions formally as follows: $\mathcal{O}_a$ satisfies the preferences of $\mathcal{O}_b$ if and only if,

$$\forall k \;\; \alpha_k^a \geq \beta_l^b \delta_l^k \tag{3.4}$$

where $k \in [1, |\mathbf{E}|]$, $l \in [1, |\mathbf{F}|]$. The above condition expression is simply checking whether $\mathcal{O}_a$ has a feature e_k (a-feature) if $\mathcal{O}_b$ has the b-feature f_l and f_l prefers e_k (b-preference).

Similarly, $\mathcal{O}_b$ satisfies the preferences of $\mathcal{O}_a$ if and only if,

$$\forall l \;\; \beta_l^b \geq \alpha_k^a \gamma_k^l \tag{3.5}$$

Now, from the relevance hypothesis we know that any pair of objects $\mathcal{O}_a$, $\mathcal{O}_b$ is relevant if and only if the bi-directional preferences are satisfied. That is,

$$\forall k \;\; \alpha_k^a \geq \beta_l^b \delta_l^k \;\; \text{and} \;\; \forall l \;\; \beta_l^b \geq \alpha_k^a \gamma_k^l \tag{3.6}$$

We assume that the relevance is *binary* and transform the above relevance condition into the following algebraic equation,

$$relevance(\mathcal{O}_a, \mathcal{O}_b) = r_{ab} = \prod_{k=1}^{K} \prod_{l=1}^{L} \underbrace{\left(\left[1 - (1 - \alpha_k^a)\beta_l^b \delta_l^k\right] \left[1 - (1 - \beta_l^b)\alpha_k^a \gamma_k^l\right] \right)}_{\text{(k, l) property preference signal}} \tag{3.7}$$

Note that Eq. 3.7 is one form of algebraic representation of bi-directional preference conditions described in E.q. 3.6. We can write it in different expressions. The first component of the Eq. 3.7 is simply checking the directional preference condition that if $\mathcal{O}_b$ is described by feature β_l^b and feature l prefers feature k ($\delta : l \rightarrow k$ ($\delta_l^k = 1$) then the $\mathcal{O}_a$ must be described by α_k^a. In other words, if $\beta_l^b = 1$ and $\delta_l^k = 1$ then $\left[1 - (1 - \alpha_k^a)\beta_l^b \delta_l^k\right] = 1$ if and only if $\alpha_k^a = 1$. Similarly, the second component is checking the other directional preference. In Eq. 3.7, if $r_{ab} = 1$ then the bi-directional conditions are satisfied, i.e., the value of (k, l) signal is "1" for each combination of (k, l). And, $r_{ab} = 0$ if $\exists$ (k, l): (k, l) signal $= 0$. In the following section, we discuss various properties of this ranking function that solves our information matching problem through a toy example.

3.4.2 Discussion

In this section, we illustrate how the theory works in estimating the relevance using a simple two users and two movies dataset. Let the users in our dataset be $\mathcal{A} = \{Emily, John\}$ and the movies set be

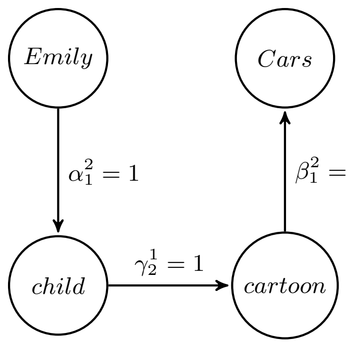


Figure 3.2: $Emily$ preference for $Cars$.

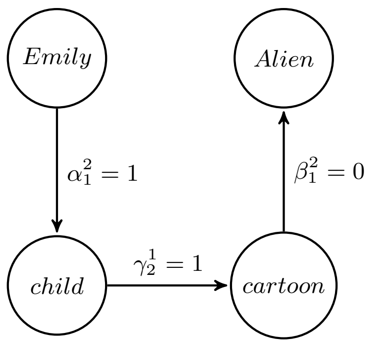


Figure 3.3: $Emily$ preference for $Alien$.

Figure 3.4: The model representations for verifying $Emily$'s preference for movie $Cars$ and $Alien$ through her elite property $child$.

$\mathcal{B} = \{Alien, Cars\}$. In order to explain the model, assume that there are two features that describe the users and two features that describe any given movie, i.e., $|\mathbf{E}| = |\mathbf{F}| = 2$. Let $\mathbf{E} = \{adult, child\}$, $\mathbf{F} = \{cartoon, horror\}$ be the user and movie features respectively. Note that $\mathbf{E}$ and $\mathbf{F}$ are two distinct sets of features. Now, the model represents each user over a binary elite a-feature space ($\mathbf{E}$) and each movie over a binary b-feature space ($\mathbf{F}$). We make further assumptions that $John$ is an adult and $Emily$ is a child. From the assumption, $John$ and $Emily$ are represented over $\mathbf{E}$ with binary vectors $\langle 1, 0\rangle$ and $\langle 0, 1\rangle$ respectively as shown in Table 3.1. Similarly, Table 3.2 represents the movies representation over $\mathbf{F}$. Let us assume that the model has access to both a-preference and b-preference functions and they are represented as follows,

$$\mathbf{G} = \begin{array}{c} \\ adult\ (e_1) \\ child\ (e_2) \end{array} \begin{array}{c} \begin{array}{cc} cartoon\ (f_1) & horror\ (f_2) \end{array} \\ \left(\begin{array}{cc} 0\ (\gamma_1^1) & 1\ (\gamma_1^2) \\ 1\ (\gamma_2^1) & 0\ (\gamma_2^2) \end{array} \right) \end{array} \quad \mathbf{H} = \begin{array}{c} \\ cartoon\ (f_1) \\ horror\ (f_2) \end{array} \begin{array}{c} \begin{array}{cc} adult\ (e_1) & child\ (e_2) \end{array} \\ \left(\begin{array}{cc} 0\ (\delta_1^1) & 1\ (\delta_1^2) \\ 1\ (\delta_2^1) & 0\ (\delta_2^2) \end{array} \right) \end{array}$$

Now, in order to compute the relevance between a given user and movie, the model verifies whether a movie satisfies the preferences of the user or not and vice-versa by representing their preferences as a function of their elite features and the preferences of the elite features. For example, any movie that satisfies $Emily$ preferences must have a feature $cartoon$ because $child$ is an elite property of $Emily$ and $child$ prefers a movie with feature $cartoon$ (from $\mathbf{G}$). The model representations in Figure 3.5 illustrate the model verification of $Emily$ preferences. In Figure 3.2, we can see that $Cars$ satisfies the preferences of $Emily$ as it has the feature $cartoon$ preferred by $Emily$'s elite feature $child$ whereas from Figure 3.3 we can see that $Alien$ does not as it does not satisfy the $child$ preferences.

Similarly, the model can verify the preferences of movies $Cars$ and $Alien$ for users $Emily$ and $John$. In the movies case, in our example, $Emily$ satisfies the preferences of $Cars$ whereas $John$ satisfies the preferences of $Alien$.

By substituting our example data representation parameter values into Eq 3.7, we can compute the relevance between user-movie pairs. For example, in order to compute $relevance(\text{Emily}, \text{Cars})$, we first compute all (k, l) feature signal property values and multiply them to get the final binary relevance.

	Alien ($\mathcal{O}_b^1$)	Cars ($\mathcal{O}_b^2$)
John ($\mathcal{O}_a^1$)	1 (r_{11})	0 (r_{12})
Emily ($\mathcal{O}_a^2$)	0 (r_{21})	1 (r_{22})

Table 3.3: Relevance between the users and movies under the model

Using our data, we can compute the $(1,1)$ feature signal value for $relevance(\text{Emily}, \text{Cars})$ as follows,

$$\left[1-(1-\alpha_1^2)\beta_1^2\delta_1^1\right] \times \left[1-(1-\beta_1^2)\alpha_1^2\gamma_1^1\right] = \left[1-\big((1-0)\times 1\times 0\big)\right] \times \left[1-\big((1-1)\times 0\times 0\big)\right] = 1$$

The resulting relevance between our example users and movies is given in Table. 3.3 where $Alien$ is relevant to $John$ whereas $Cars$ is relevant to $Emily$. We can intuitively see this relevance by simply observing the orthogonal nature of the example data. To illustrate an interesting property of the theory, consider a scenario where we think both $cartoon$ and $horror$ are elite properties of $Cars$ and we know the preference function matrices $\mathbf{G}$ and $\mathbf{H}$. Now, let us say we observed that $Cars$ is relevant to $Emily$. From our relevance function and bi-directional preferences, we get $relevance(Emily, Cars) = 1$ if and only if $horror$ does not describe $Cars$ but $cartoon$ does. By using observed relevance information and preference functions, we can modify the $Cars$ feature representation by setting $horror$ as its non-elite property. Similarly, at the same time, we can also modify the feature representation of $Emily$ using preference functions and the observed relevance. This essentially means that the theory allows us to modify both information objects independently at the same time in light of new relevance information. This is exactly the solution needed to solve the original problem of the unified model [RMC82a, Rob03] and one of the requirements for solving our information matching problem. Note that, we can modify the preference functions based on the relevance.

Another property of the theory is that, we are able to represent two different types of information objects in two distinct feature spaces. This allows us to model each object with a different set of independent features, unlike standard matching solutions in collaborative filtering or ad-hoc retrieval where both the objects have to be represented in a common feature space. The primary advantage being that we can add distinct features to information objects without affecting the core matching model.

Finally, the preference functions $\mathbf{G}$ and $\mathbf{H}$ represent the global preferences between features that describe two distinct types of information objects. These preferences are independent of individual objects and represent the preferences of features shared by a collection of (similar) objects. So, when we are verifying the preferences of an object through $\mathbf{G}$ or $\mathbf{H}$, we are indeed using the similar objects' information through $\mathbf{G}$ or $\mathbf{H}$. This is an interesting property as we are able to use similar objects' information to a given object without computing any explicit similarity between the objects as opposed to the model developed by Wang et.al. [WdVR06]. This is another requirement of a solution to our problem and our theory is able to incorporate it into relevance matching. All the above properties illustrate that our theory is a solution for the problem of information matching. In the following section, we develop a probabilistic model based on the theory presented in this section.

3.5 Probabilistic Model

In the previous section we presented a new theory and model based on information matching and bi-directional relevance hypothesis. The fundamental principles behind the idea are (a) that we represent any given object with a set of binary properties and (b) that we assume relevance is only dependent on bi-directional preferences of objects represented through their elite properties. From the observation, to determine the relevance, we need to know the binary representation of objects over feature spaces $\mathbf{E}$, $\mathbf{F}$ and matrices $\mathbf{G}, \mathbf{H}$. But in practice, it is impossible to determine the exact representations and matrices. So, an obvious way to solve this problem is by introducing uncertainties in both elite properties of the objects, and in the preferences and then estimating them. Once the representation is probabilistic then there will be uncertainty in relevance. So, the objective of matching will be to find the probability of *relevance/match* between two objects.

Another important aspect of the theory is that all the properties are binary, and the assumption is that if we know all the elite properties of information objects, it is sufficient to determine the relevance using the preference functions. But, in practical systems, we observe a variety of features that are not binary, e.g., terms with term frequency in a document object. We can handle this by defining a binary elite property for each feature and defining a generative model to generate observed feature value based on the eliteness of the property. For example, in our text document objects, we can define an elite property for each term in the vocabulary and define a generative model to generate the observed term frequency in a document based on the eliteness of the property for that document object.

In order to develop the probabilistic version of the model we need the probabilistic representation of (a) information objects, (b) preference mappings and (c) relevance. We start by defining some notation to represent information objects. Let $\mathbf{X}_m$ be the vector of observed feature values for m^{th} object of type $\mathcal{A}$ or $\mathcal{O}_a$ where $\mathbf{X}_m = \{x_1^m, x_2^m, \cdots, x_K^m\}$. For example x_k^m would be the k^{th} term's term frequency in the m^{th} document. Similarly, let $\mathbf{Y}_n$ be the vector of observed feature values for n^{th} object of type $\mathcal{B}$ or $\mathcal{O}_b$ where $\mathbf{Y}_n = \{y_1^n, y_2^n, \cdots, y_L^n\}$. Now, let $\mathbf{e}^m$ and $\mathbf{f}^n$ be the binary feature vector representation of objects $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ over feature spaces $\mathbf{E}$ and $\mathbf{F}$ respectively where $\mathbf{e}_m = \langle \alpha_1^m, \cdots, \alpha_K^m \rangle$ and $\mathbf{f}_n = \langle \beta_1^n, \cdots, \beta_L^n \rangle$.

In order to model probabilistically, we define three generative processes for observed information, i.e., information objects and relevance, as follows:

1. Any given information object $\mathcal{O}_a^m (or \mathbf{X}_m)$ is generated by choosing a binary vector from $\{0, 1\}^{|\mathbf{E}|}$, i.e., $\mathbf{e}_m$. In other words, we choose a set of elite properties from $\mathbf{E}$ feature space and generate the observed $\mathbf{X}_m$.

2. Similarly, any given information object $\mathcal{O}_b^n (or \mathbf{Y}_n)$ is generated by choosing a binary vector from $\{0, 1\}^{|\mathbf{F}|}$, i.e., $\mathbf{f}_n$.

3. The binary relevance between $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ denoted by r_{mn} is generated from the bi-directional preference matching function of $\mathbf{e}_m, \mathbf{f}_n$ and the preference functions $\mathbf{G}$ and $\mathbf{F}$.

In the following, we first describe each generative model independently and combine at the end into a single model. We start with the probabilistic representation of the r_{mn}.

Generating relevance

We know that the r_{mn} is generated by the binary elite properties of information objects and the preference function of those elite properties. We write the likelihood of r_{mn} using the relevance function from E.q. 3.7 as follows,

$$p(r_{mn}|\mathbf{e}_m, \mathbf{f}_n, \mathbf{G}, \mathbf{H}) = \prod_{k=1}^{K}\prod_{l=1}^{L}\left(\left[1-(1-\alpha_k^m)\beta_l^n\delta_l^k\right]\left[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\right]\right)^{(r_{mn})}$$
$$\left(1-\left(\left[1-(1-\alpha_k^m)\beta_j^n\delta_l^k\right]\left[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\right]\right)\right)^{\left(1-r_{mn}\right)}$$

An important note here is, we implicitly made an assumption that α_k^m, β_l^n , γ_k^l and δ_l^k values are independent of all other property values in $\mathbf{e}_m, \mathbf{f}_n$, $\mathbf{G}$ and $\mathbf{H}$ respectively. The assumption is purely to make the computation possible. And, the probability of relevance, r_{mn}, is defined over the cross product of four different structured probability spaces, namely $\mathbf{E}, \mathbf{F}, \mathbf{E} \times \mathbf{F}$ $(\mathbf{G})$ and $\mathbf{F} \times \mathbf{E}$ $(\mathbf{H})$. The definition of relevance over a structured probability space allows the model to consistently modify the probability in four distinct spaces based on observed relevance. That is, we can modify the object representation based on relevance in $\mathbf{E}$ (e.g., modifying document representation), $\mathbf{F}$ (e.g., modifying query representation) and preferences mappings in $\mathbf{G}$ and $\mathbf{H}$ (i.e., modifying the relevance feature mappings applicable to all the objects that share the same properties (similar documents and queries)). This is precisely the requirement of the problem of the probabilistic unified model [Rob03].

The $\alpha_k^m, \beta_l^n, \gamma_k^l$ and δ_l^k values are binary and hidden as we don't know the exact values of vectors $\mathbf{e}_m, \mathbf{f}_n$ and matrices $\mathbf{G}$ and $\mathbf{H}$. We therefore assume each of these latent variables follow a beta distribution as follows,

$$\alpha_k^m \sim \mathcal{B}(\theta^1_{\alpha_k^l}, \theta^2_{\alpha_k^l}) = {\alpha_k^l}^{\theta^1}(1-\alpha_k^l)^{\theta^2_{\alpha_k^l}} \qquad \beta_l^n \sim \mathcal{B}(\theta^1_{\beta_l^n}, \theta^2_{\beta_l^n}) = {\beta_l^n}^{\theta^1_{\beta_l^n}}(1-\beta_l^n)^{\theta^2_{\beta_l^n}}$$
$$\gamma_k^l \sim \mathcal{B}(\theta^1_{\gamma_k^l}, \theta^2_{\gamma_k^l}) = {\gamma_k^l}^{\theta^1}(1-\gamma_k^l)^{\theta^2_{\gamma_k^l}} \qquad \delta_l^k \sim \mathcal{B}(\theta^1_{\delta_l^k}, \theta^2_{\delta_l^k}) = {\delta_l^k}^{\theta^1_{\delta_l^k}}(1-\delta_l^k)^{\theta^2_{\delta_l^k}}$$

where θ's are hyper parameters of the distributions.

Generating information objects

The probability of observing an object $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ is given by

$$p(\mathcal{O}_a^m) = p(\mathbf{X}_m) = \sum_{i=1}^{2^{|\mathbf{E}|}} p(\mathbf{X}_m|\mathbf{e}_i)p(\mathbf{e}_i) \quad \text{and} \quad p(\mathcal{O}_b^n) = p(\mathbf{Y}_n) = \sum_{j=1}^{2^{|\mathbf{F}}} p(\mathbf{Y}_n|\mathbf{f}_j)p(\mathbf{f}_j) \tag{3.8}$$

To simplify Eq. 3.8 and make it tractable, we make the following assumption: the observed features of an object are independent of each other. That is,

$$p(\mathbf{X}_m) = \prod_{k=1}^{K} p(x_k^m) \quad \text{and} \quad p(\mathbf{Y}_n) = \prod_{l=1}^{L} p(y_n^l) \tag{3.9}$$

From our generative process, we know that the observed x_k^m is generated by the eliteness value of α_k^m for $\mathcal{O}_a^m$ (α_k^m). Therefore, the observed x_k^m value can be thought of as drawn from a two stage generative

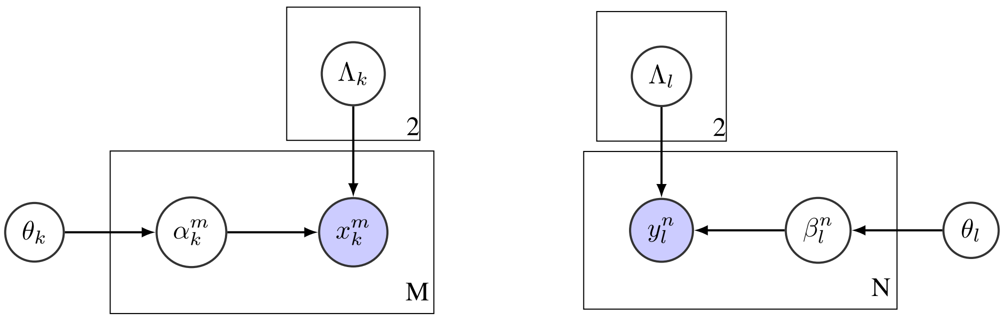


Figure 3.5: The graphical model representation of the 2-Class mixture model where x_k^m and y_l^n are the observed k^{th}, l^{th} feature values for the objects $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ respectively. The Λ's are corresponding feature's mixture distributions, α_k^m and β_l^n are binary eliteness values and θ's are priors.

process: (a) choose α_k^m for $\mathcal{O}_a^m$, i.e., α_k^m; then, choose x_k^m from the distribution $p(x_k^m|\alpha_k)$. As α_k^m is binary, we assume that $p(x_k^m|\alpha_k = 1)$ follows one distribution and $p(x_k^m|\alpha_k = 0)$ follows another distribution. In other words the eliteness value of a property to a given object stochastically determines the corresponding feature's observed value in an object.

Now, from the above generative model $p(x_k^m)$ is given by

$$p(x_k^m) = p(\alpha_k^m = 1)p(x_k^m|\alpha_k = 1) + p(\alpha_k^m = 0)p(x_k^m|\alpha_k = 0) \tag{3.10}$$

Note that we substituted α_k^m with binary values. The graphical models for generating x_k^m and y_l^n are given in Figure. 3.5. The joint probability density of the observed data and parameters, from the graphical models, is given by

$$p(\mathcal{D}^{x_k}, \{\Lambda_k^0, \Lambda_k^1\}, \theta_k) = \prod_{m=1}^{|\mathcal{A}|} \Big(p(x_k^m|\alpha_k^m, \{\Lambda_k^0, \Lambda_k^1\})p(\alpha_k^m|\theta_k)\Big)p(\Lambda_k^0)p(\Lambda_k^1)p(\theta_k), \tag{3.11}$$

and

$$p(\mathcal{D}^{y_l}, \{\Lambda_l^0, \Lambda_l^1\}, \theta_l) = \prod_{n=1}^{|\mathcal{B}|} \Big(p(y_l^n|\beta_l^n, \{\Lambda_l^0, \Lambda_l^1\})p(\beta_l^n|\theta_l)\Big)p(\Lambda_l^0)p(\Lambda_l^1)p(\theta_l), \tag{3.12}$$

where $M = |\mathcal{A}|$, $N = |\mathcal{B}|$, $\mathcal{D}^{x_k} \overset{def}{=} \{x_k^m\}_{m=1}^{|\mathcal{A}|}$, $\mathcal{D}^{y_l} \overset{def}{=} \{y_l^n\}_{n=1}^{|\mathcal{B}|}$ and Λ_k^0, Λ_k^1, Λ_l^0 and Λ_l^1 are parameters of the mixture distributions.

Using the generative models specified in the graphical model in Figure. 3.5, if we have $\mathbf{X}_m$ and $\mathbf{Y}_n$, we can infer each component of x_k^m and y_l^n by assuming one distribution for $p(x_k^m|\alpha_k^m = 1)$ and another distribution for $p(x_k^m|\alpha_k^m = 0)$. These distributions can be modelled based on the type of data we observe. And, there could be different features modelled with different distributions. For example if the observed values ($\mathbf{x}_k^m$) are term frequencies in a document collection, then we can assume that the term frequency of k^{th} term, x_k^m, in any given document, $\mathcal{O}_a^m$ is a sample from a Poisson distribution with rate λ_1 if $\alpha_k^m = 1$, otherwise a sample from another Poisson distribution with rate λ_0. Using a set of term frequency observations of the term over the document collection, we can estimate the λ rate for each distribution, and infer the α_k^m for any given $\mathcal{O}_m$. This is the assumption that is commonly used in the 2-Poisson and BM25 models for generating term frequencies in the document [RZ09].

In general terms, the model is stating (a) that every elite property/feature value follows a distribution over the objects that the feature describes and another distribution over the objects it does not describe. And, the model learns these distributions from the observed data for every feature. Also, (b) for any given object the model determines the probability of the feature being elite or non-elite by looking at the observed feature value and the distributions learned in (a). This gives us the probabilistic representation of an object over a given feature. The relevance is generated from the combination of object representations and the mappings between the elite properties based on the bi-directional preferences. We now put together all the three generative components of the model and define a single generative model as shown in the graphical model, Figure 3.6.

3.5.1 Graphical Model

The graphical model in Figure 3.6 consist of three generative processes that combine all the information in generating relevance,

1. Generation of x_k^m, $\forall\, m, k$ using α_k^m and Λ_k.
2. Generation of y_l^n, $\forall\, n, l$ using β_l^n and Λ_l.
3. Generation of r_{mn}, $\forall m, n$ using $\{(\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k)\}_{k=1,l=1}^{K,L}$.

There are four plates representing objects and features as follows: (a) the bottom two plates denote $|\mathcal{A}|, |\mathcal{B}|$ number of objects of type $\mathcal{O}_a$ and $\mathcal{O}_b$ respectively, (b) the top two plates denote a set of K, L elite properties that describe $\mathcal{O}_a$ and $\mathcal{O}_b$ respectively, and (c) the overlap between the features plates represents the mapping between the features of two distinct types of objects, i.e., γ_k^l and δ_l^k.

The variable $\hat{r}_{mn}$ is the estimated relevance between $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ whereas r_{mn} is the observed relevance or judge assigned relevance. We assume noise over the observed relevance and model the relevance as a combination of expected relevance and the noise $\mathbf{J}$. $\mathbf{J}$ is crucial for incorporating the inconsistencies in relevance assessments given by the judges. Finally, $\mathbf{z}_{mn}^{kl}$ is the vector of latent variables indicating the k, l property signal in Eq. 3.7 where $\mathbf{z}_{mn} = \{z_{mn}^{11}, \cdots, z_{mn}^{kl}\}$ and $z_{mn}^{kl} \in \{0, 1\}$. Each component of $\mathbf{z}_{mn}$, z_{mn}^{kl}, denotes a binary value representing whether the bi-directional preferences of feature k and l are satisfied or not for a pair of objects $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$. Finally, $r_{mn} = 1$ if and only if $\forall (m, n)\; z_{mn}^{kl} = 1$, i.e., the bi-directional preferences are satisfied for features that describe both objects.

The joint density

In order to obtain a method to perform the approximated inference in the graphical model, we write the joint density of the model in the following way. First, we denote by $\mathbf{\Theta}$ the set of all parameters where

$$\mathbf{\Theta} = \Big\{\{\alpha_1^m, \cdots \alpha_k^m\}_{m=1}^{|\mathcal{A}|}\, \{\beta_1^n, \cdots \beta_l^n\}_{n=1}^{|\mathcal{B}|}, \{\gamma_1^1, \cdots \gamma_k^l\}_{k=1,l=1}^{K,L}, \{\delta_1^1, \cdots \delta_l^k\}_{k=1,l=1}^{K,L}\Big\}$$

and denote hyper parameters by $\mathcal{H}$. Let $\mathcal{D}$ be the set of all observations, i.e., $\big(\mathcal{D}^r, \mathcal{D}^m, \mathcal{D}^n\big) \in \mathcal{D}$. Now, the joint probability density of observing data and parameters given $\mathcal{H}$ from the graphical model is given by

$$p(\mathcal{D}, \mathbf{\Theta}|\mathcal{H}) = \prod_{m=1}^{M} \prod_{n=1}^{N} \Bigg[p(r_{mn}|\hat{r}_{mn}, \mathbf{J}) p(\hat{r}_{mn}|\mathbf{z}_{mn}) \prod_{k=1}^{K} \prod_{l=1}^{L} p(z_{mn}^{kl}|\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k)$$

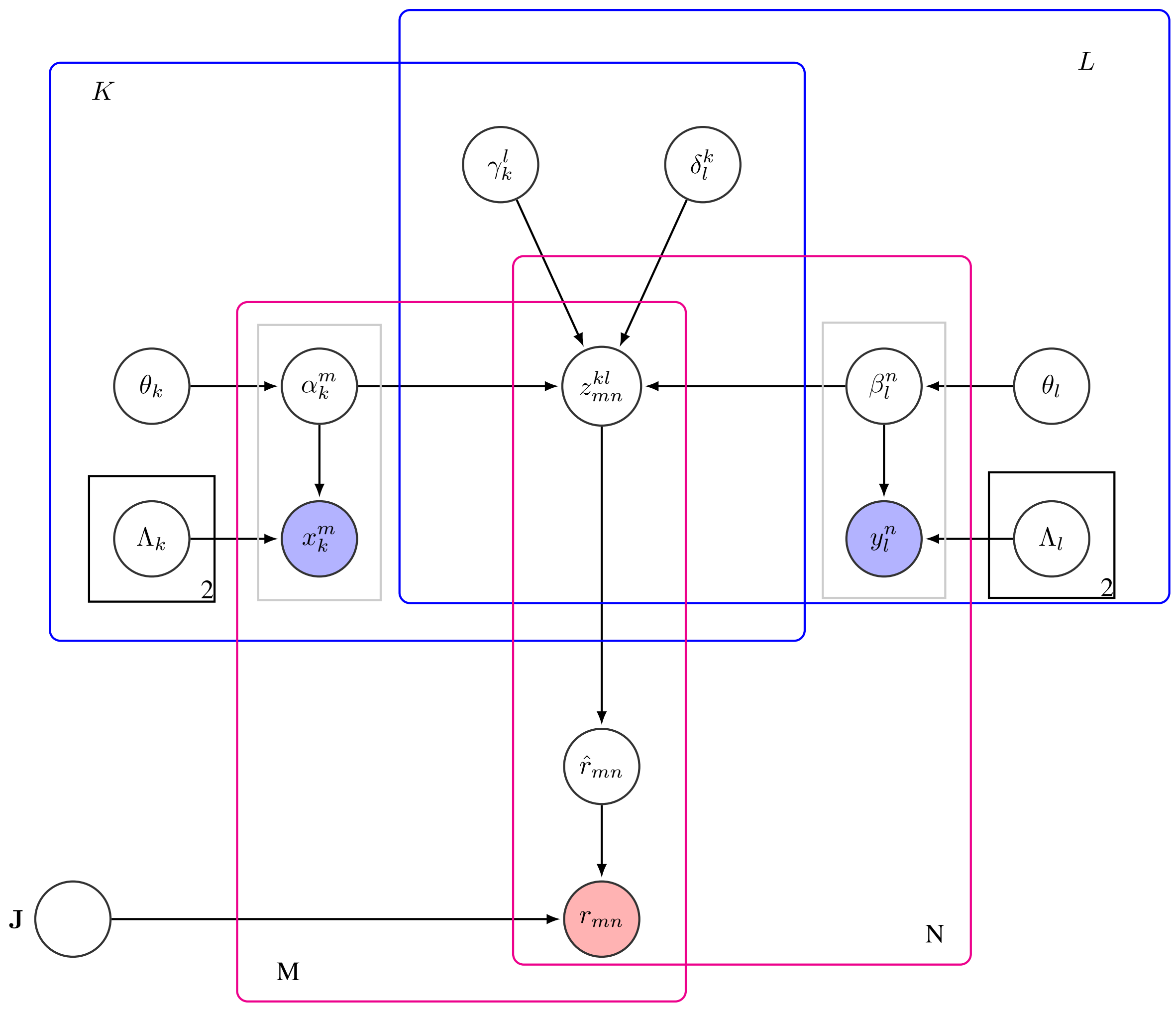


Figure 3.6: The graphical model describing the probabilistic generative model based on the theory of information matching.

$$\cdot \prod_{k=1}^{K} p(x_k^m|\alpha_k^m, \Lambda_k) \prod_{l=1}^{L} p(y_l^n|\beta_l^n, \Lambda_l)\Bigg] \cdot \prod_{m=1}^{M}\prod_{n=1}^{N}\left[\prod_{k=1}^{K} \mathcal{B}(\alpha_k^m\,;\,\Theta^{\alpha_k^m}) \prod_{l=1}^{L} \mathcal{B}(\beta_l^n\,;\,\Theta^{\beta_l^n})\right]$$
$$\cdot \prod_{k=1}^{K}\prod_{l=1}^{L} \mathcal{B}(\gamma_k^l\,;\,\Theta_k^l)\mathcal{B}(\delta_l^k\,;\,\Theta_l^k) \cdot \prod_{k=1}^{K} p(\Lambda_k^0)p(\Lambda_k^1) \prod_{l=1}^{L} p(\Lambda_l^0)p(\Lambda_l^1),$$

where

$$p(z_{mn}^{kl}|\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k) = \Big(\big[1-(1-\alpha_k^m)\beta_l^n\delta_l^k\big]\big[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\big]\Big)^{(z_{mn}^{kl})}$$
$$\Big(1-\Big(\big[1-(1-\alpha_k^m)\beta_j^n\delta_l^k\big]\big[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\big]\Big)\Big)^{\left(1-z_{mn}^{kl}\right)}$$

In order to estimate the parameters of $\mathbf{\Theta}$, we use Bayes' theorem to infer the posterior density of $\mathbf{\Theta}$,

$$p(\mathbf{\Theta}|\mathcal{D},\mathcal{H}) = \frac{p(\mathcal{D}|\mathbf{\Theta})p(\mathbf{\Theta}|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})} \tag{3.13}$$

The direct computation of posterior distribution $p(\boldsymbol{\Theta}|\mathcal{D},\mathcal{H})$ is computationally intractable [Mac02]. Hence, we estimate the posterior density parameters with approximate inference techniques. We provide the optimisation procedure for parameter estimation in Chapter 4. In the subsequent chapters, we also refer the probabilistic model based on the *unified information matching framework* as the *Bi-directional unified model* since the relevance is computed based on directional preference matching. Additionally, the model is capable of incorporating/modelling the information from multiple sources into a single probabilistic relevance model.

3.6 Discussion

From the above graphical model, we can infer α_k^m and β_l^n using observed x_k^m and y_l^n respectively. That is probabilistically representing the objects using observed feature values. And, once the r_{mn} is observed we can update the belief on α_k^m and β_l^n through $\mathbf{z}_{mn}$ as r_{mn} is generated from $\mathbf{z}_{mn}$. In other words, through $\mathbf{z}_{mn}$ we can modify the $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ representation over elite space. Through the model, we are able to modify the representations of both objects based on the observed relevance. For example, assume that $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$ are a document-query pair and $\mathbf{X}_m$ and $\mathbf{Y}_n$ are vectors of term frequencies corresponding to vocabulary features. Now, we can infer the eliteness value of α_k, β_l using x_k^m, y_l^n. But once we observe the relevance between the document-query pair we can modify both the document and query eliteness probabilities of each elite property. The model is able to use the relevance information on both document *and* query, and modify the representations of information objects.

Similarly, once the relevance is observed on a pair of objects, the model can update the preferences function probabilities, γ_k^l and δ_l^k, through $\mathbf{z}_{mn}$. What this means is that the modification of the probabilities of γ_k^l and δ_l^k will affect the relevance between any other objects that share the properties k and l, i.e., similar objects of $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$, as the relevance is dependent on the preference functions as well as object representations. Therefore, the model is able to use the relevance information on one pair of objects and influence the relevance of other similar objects (or objects with shared properties). In other words, when we are computing the relevance between two objects $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$, the model is using the individual objects information through $\mathbf{X}_m$ and $\mathbf{Y}_n$, relevance information on both $\mathcal{O}_a^m$ and $\mathcal{O}_b^n$, and relevance information on other similar objects through γ_k^l and δ_l^k. In conclusion, our model is able to unify all the available information in computing the probability of relevance between two objects.

An important aspect of this probabilistic model is that we can represent the objects with the features of our choice as there is no restriction on the feature set for representing any object. This is contrary to other models where you need a common feature space, either hidden or observed, for computing the relevance. In BM25, Language Models and generative theory of relevance, the models represent the document and query (information objects) in common vocabulary terms space [RZ09, LZ01, PC98, Lav04]. In contrast, the models in collaborative filtering and IR model based on PLSI or topic models assume a common hidden feature space [KBV09, Hof99, BNJ03]. In both cases the models assume common feature space. This assumption restricts the ability to model objects with other feature that are not common between the information objects (document and query), and the relevance is defined only on the common feature space. Our model circumvents this problem due to the bi-directional relevance hypothesis. Another important aspect of the model is that we do not need to compute explicit similarity between objects to use similar objects' information in computing the relevance as in other collaborative filtering and unified collaborative filtering models [DK04, WdVR06].

In this chapter, we presented a new theory and probabilistic relevance model based on the information matching model hypothesis. The hypothesis is defined based on the ideas of bi-directional preference matching. The resulting model represents the information objects in two different feature spaces thus giving us the freedom to model the objects with any features without restrictions. The model also lets us modify the original objects representation over the corresponding elite space based on the observed relevance thus, solving the long standing problem of the unified model [RMC82a]. The ideas presented here can be used to solve/applied to number of problems in information matching without any alteration. Some of the common problems we can tackle using this model are: (a) Recommendation (b) Group Recommendation and (c) Text Retrieval. In the following chapter, we present how this model can be applied to the above mentioned three problems.

Chapter 4

Applications & Parameter Estimation

In Chapter 3, we presented the theory of information matching and its formulation in a generic way. The main objective of this chapter is to show how we can apply our theory of information matching to three retrieval scenarios where, in the past, each scenario is being addressed with a different solution framework. In order to achieve our objective, we derive relevance ranking functions for three different retrieval scenarios from our single theory. Before we describe and derive ranking functions for popular retrieval scenarios, let us consider a few possibilities of the problem of matching depending on the availability of various kinds of information. In one case, we may have only the interaction between two types of information objects, i.e. the available data, $\mathcal{D}$, for the retrieval system is some observation on $\{\mathcal{A} \times \mathcal{B}\}$. In IR problems, this scenario describes a retrieval system having access to a set of ratings corresponding to the interaction between a set of user and item pairs or relevance judgements between document and query pairs without having access to any other information. The most popular retrieval problem corresponding to this scenario is described as the problem of collaborative filtering. A second case, we may have access to the interaction between the objects as described in the previous case along with additional contextual information on the given set of information objects. An increasingly popular retrieval scenario corresponding to this case is the problem of group recommendation, where the retrieval system has to recommend a set of items to a group of individuals. In order to do this, the system has access to a set of ratings corresponding to a set of user-item pairs and contextual information describing the association of each individual user to a particular group. Consider a third scenario which has no information available on interaction between the objects but has the information about the individual objects. The most common retrieval scenario corresponding to this case is the problem of ad-hoc text retrieval with no relevance information available to the system. Finally, we may have access to a combination of some or all the above described scenarios. In the following sections we show that our theoretical model of information matching can be applied to all the above scenarios.

During the course of this chapter, we will address the following three popular retrieval scenarios:

1. **Collaborative Filtering (Recommendation):** The objective of collaborative filtering (recommendation) is to *recommend* a set of *relevant* items (movies, books, etc.) for a given user. The data available to compute these relevant items is the *preferences/tastes* of users represented in the form of their ratings on a set of items.

In this particular problem, there are no common attributes (information space) between the users and items unlike the document-query pairs where they share a common vocabulary. To apply our model, we model user and item as two distinct information objects and treat the user-item rating (with some threshold) as a relevance between these information objects. In other words, we consider all the items that a user *liked* are the relevance judgements of the user and vice-versa. This application would serve as a demonstration of the model (a) for utilising the relevance information on both information objects at the same time in computing the probability of relevance (b) that it neither represents the user and item in a common feature space nor computes explicit similarity between information objects in computing the relevance.

2. **Group Recommendation:** The problem is to recommend a set of relevant items to a group of individuals. This application has extra contextual information to be considered while computing relevant items for a given group of individuals. The data available to a retrieval system in this scenario is a set of user-item ratings representing the preferences of individual members and the group information specifying the individuals belonging to a group. Now, the objective is to compute a set of relevant items for a group given that we know the individual and group preferences. Later, we will discuss the problems with existing popular approaches used to tackle this problem using simple examples, and show why we need a model (like ours) that is capable of unifying both contextual information (group) along with individual members' preferences into a single model.

   This application will serve as a demonstration of our theory to combine extra contextual information along with information about two distinct information objects (user and item) and their interactions (ratings) in a single unified relevance model. The group recommendation problem also includes recommending to a group of individuals where we know the preferences of a group but not its individual members' preferences. For example, a home digital TV or Xbox console where a single device is used by multiple members of the family (e.g. children, house wife, old members of the family). Each individual in the family might have different preferences on what they like to watch/play. However, in the system, we see the preferences of all members as a single user. We will not address this problem though it can be tackled with the theory of our model!

3. **Ad-hoc text retrieval:** In general, the problem can be stated as follows: we have a set of documents (e.g., written in English) each representing some information, and a query (relatively short in length compared to document) representing an information need. The objective is to retrieve a set of documents *relevant* to the query. The data available to the retrieval system is the set of vocabulary terms in the document collection (and by implication their statistics) and query. In addition, the system may have access to a small set of human relevance judgments of query-document pairs. The model should be applicable in both the presence and absence of relevance judgments.

   This application will demonstrate that our model can be applied in general text retrieval tasks where the features of both query and documents are derived from a common vocabulary, and when relevance information is not available to the retrieval system.

In addition to the above three scenarios, we also describe the complete model when more than one of the above scenarios are present in a matching problem. In chapter 5, we demonstrate this model through a practical system. As of now there is no single relevance ranking model that is shown to be applied to both ad-hoc retrieval and recommendation. This is the first successful model, to our knowledge, that seamlessly addresses both as the underlying problem is matching two information objects. Finally, in this chapter, we derive parameter estimation methods for learning the parameters in our model for subsequent use in the following chapter for experimentation.

As stated, our principle objective is to demonstrate the application of the theory of information matching to different retrieval scenarios and their conceptual relation to other models/frameworks both theoretically and experimentally. Therefore, we follow traditional approaches in deriving the relevance ranking functions under the *theory of information matching* by doing the following: (1) we define a single binary relevance random variable r as opposed to defining a relevance random variable for each pair as in chapter 3. We denote the relevance by $r = 1$ if both objects are relevant, and 0 otherwise (2) We infer the model parameters using Empirical Bayes as opposed to complete Bayesian inference. However, we provide a fully Bayesian inference solution at the end of the chapter.

In the following sections, for each of the above mentioned scenarios, we do the following: (a) we formally describe the problem and represent it with the elements of our *theory of information matching*, (b) derive the relevance ranking function, and (c) provide some discussion on the application.

4.1 Relevance under bi-directional preferences

Before we get into each scenario and provide a relevance ranking function, we would like to provide some analysis on relevance function in our information matching model. Let's start by looking at the following relevance function described in page 46 of the Chapter 3,

$$relevance(\mathcal{O}^m, \mathcal{O}^n) = r_{mn} = \prod_{k=1}^{K}\prod_{l=1}^{L} \underbrace{\Bigg(\big[1 - (1-\alpha_k^m)\beta_l^n \delta_l^k\big]\big[1 - (1-\beta_l^n)\alpha_k^m \gamma_k^l\big] \Bigg)}_{\text{(k, l) bi-directional preference signal } (z_{mn}^{kl})} \tag{4.1}$$

The two information objects $\mathcal{O}^m$ and $\mathcal{O}^n$ are relevant if and only if $\forall_{k,l} z_{mn}^{kl} = 1$. In other words, the directional preference conditions of all the elite properties of both $\mathcal{O}^m$ and $\mathcal{O}^n$ must be satisfied. By close observation of Eq. (4.1), we can state that if we know that the $\mathcal{O}^m$ and $\mathcal{O}^n$ pair is relevant, i.e., $r_{mn} = 1$, then

$$\forall_{k,l} z_{mn}^{kl} = 1 \quad \text{where} \quad z_{mn}^{kl} = \Bigg(\big[1 - (1-\alpha_k^m)\beta_l^n \delta_l^k\big]\big[1 - (1-\beta_l^n)\alpha_k^m \gamma_k^l\big] \Bigg) \tag{4.2}$$

Now, let's find out the binary values of feature parameters when $z_{mn}^{kl} = 1$ and $z_{mn}^{kl} = 0$. Table 4.1 presents the z_{mn}^{kl} value for all the combinations of elite properties' values of an example with one feature describing each object (a user and a movie). From the table, we can deduce the following logical relations:

$$z_{mn}^{kl} = 0 \Longrightarrow \Big(\big((1-\alpha_k)\beta_l \delta_l^k\big) \vee \big(\alpha_k(1-\beta_l)\gamma_k^l\big) \Big) \tag{4.3}$$

$child\ (k)$	$cartoon\ (l)$	$child \rightarrow cartoon$	$cartoon \rightarrow child$	z_{mn}^{kl}
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1
1	0	0	1	1
1	0	0	0	1
0	1	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	1	0	0
1	0	1	1	0

Table 4.1: The bi-directional preference signal value for all possible combinations of values of elite properties, and their preferences in relevance function for a given user-movie and user-movie feature pairs. In this example, the user feature set is $\{child\}$ and movie feature set is $\{cartoon\}$ for a given m^{th} user and n^{th} movie.

$$z_{mn}^{kl} = 1 \Longrightarrow \Bigg(\big((1-\alpha_k)(1-\delta_l^k)\beta_l\big) \vee \big(\alpha_k(1-\beta_l)(1-\gamma_l^k)\big) \vee (\alpha_k\beta_l) \vee \big((1-\alpha_k)(1-\beta_l)\big)\Bigg) \tag{4.4}$$

The above logical relations are a representation of whether a pair of features satisfy the bi-directional preferences with respect to two given information objects or not. Later, we will exploit the logical relationships in Eq. 4.3 and Eq. 4.4 in relation to relevance for simplifying the model parameter estimations.

4.2 Collaborative Filtering (Recommendation)

The objective of the ranking function in collaborative filtering (CF) is to rank a set of items for a given user based on their probability of relevance. However, the only information available to the retrieval models in this case is a set of users, items and some user-item ratings. Before we describe how our information matching framework and model can be applied to solve the problem of ranking in CF, we define a notation specific to the problem for the sake of simplicity. Let us assume that there are M users and N items represented by sets $\mathcal{U}$ and $\mathcal{V}$ respectively where $\mathcal{U} = \{\mathbf{u}_1, \cdots, \mathbf{u}_M\}$ and $\mathcal{V} = \{\mathbf{v}_1, \cdots \mathbf{v}_N\}$. We distinguish the users with index m and items with index n, e.g., $\mathbf{u}_m \in \mathcal{U}$ and $\mathbf{v}_n \in \mathcal{V}$. Finally, we denote by $\mathcal{D}^{rt} \overset{def}{=} \{rt_{mn}\}$ the set of user-item rating pairs. Finally, let r_{mn} be the binary relevance random variable where $r_{mn} = 1$ denotes the fact that the item $\mathbf{v}_n$ is relevant to user $\mathbf{u}_m$, and by $r_{mn} = 0$ otherwise. We note that we use the terms 'elite properties' and 'features' interchangeably.

In order to rank a set of items in $\mathcal{V}$ in the order of relevance for a given user $\mathbf{u}_m$, all we need is for all n to compute the probability $p(r_{mn} = 1|\mathbf{u}_m, \mathbf{v}_n)$ and rank $\mathbf{v}_n$'s in the decreasing order of probability. To compute $p(r_{mn} = 1|\mathbf{u}_m, \mathbf{v}_n)$, let us map this problem into elements of the *information matching framework* (IMF). In IMF representation, $\mathcal{U}$ and $\mathcal{V}$ are two sets of information objects of type $\mathcal{O}_a$ and $\mathcal{O}_b$ respectively where $\mathcal{U} \equiv \mathcal{A}$, $\mathcal{V} \equiv \mathcal{B}$, $\mathbf{u}_m \equiv \mathcal{O}_a$ and $\mathbf{v}_n \equiv \mathcal{O}_b$.

To apply the matching model, we need to:

1. Define two different sets of elite property (feature) spaces for representing users and items respectively. That is, we need to define a-features and b-features sets. For the sake of better readability, we refer to these feature sets as u-features and v-features for representing user and items respectively.

2. define preference mapping functions between a-features and b-features, i.e., a-preference and b-preference. Similarly, we call u-preference, v-appeal for representing the mappings between u-features and v-features respectively;

3. identify a set of relevant user-item pairs for the model to learn the parameters of user, item representations and preference mappings.

In the following section we describe how we define properties/features for applying our framework to a collaborative filtering problem.

4.2.1 Features (elite properties), Mappings and Relevant pairs

In Chapter 3's mathematical formulation of framework, we have represented each information object of type $\mathcal{O}_a$ ($\mathcal{O}_a \in \mathcal{A}$) over a set of binary elite properties denoted by $\mathbf{E}$. Similarly, each object of type $\mathcal{O}_b$ ($\mathcal{O}_b \in \mathcal{B}$) is represented over a set of binary elite properties denoted by $\mathbf{F}$. Where $\mathbf{E} = \{e_1, \cdots, e_K\}$, $\mathbf{F} = \{f_1, \cdots, f_L\}$ and the binary value of the component e_k for a given object $\mathcal{O}_a$ is denoted by α_k^a where $k \in [1, K]$. Similarly, the binary value of component f_l for a given object $\mathcal{O}_b$ is denoted by β_l^b where $l \in [1, L]$.

Our first objective is to define the elite property (feature) sets $\mathbf{E}$ and $\mathbf{F}$ for representing our information objects of type $\mathcal{U}$ and $\mathcal{V}$. Once we define these feature sets, we can represent each user over a user feature set as $\mathbf{u}_m \in \{0,1\}^{|\mathbf{E}|}$ and item over item feature as $\mathbf{v}_n \in \{0,1\}^{|\mathbf{F}|}$. As per our theory and unified matching framework, $\mathbf{E}$ and $\mathbf{F}$ are two distinct feature sets and defined independent of each other. In the collaborative filtering (CF) problem, the only information available to the retrieval system is the interaction between users and items either in the form of ratings or relevance judgments. As a result, there are no common properties (or features) between users and items to perform matching.

As there are no user or item specific features in CF, we create a user feature set $\mathbf{E}$ by defining a binary property corresponding to each item in our collection, and each property is labeled with k where $k = 1, \cdots, N$ ($K = |\mathbf{E}| = N = |\mathcal{V}|$). In other words, in the absence of any other properties/features, the 'features' of users are associated with individual items – e.g. 'this is an example of the kind of item that I like'. The k^{th} feature in $\mathbf{E}$ corresponds to a "kind of item" $\mathbf{v}_k$. For example, in a simple movie recommendation, the user feature set $\mathbf{E}$ is a set of movies. And, each binary feature value for a given user corresponding to each feature represents whether that user likes the 'kind of movie' the feature represents. Note that, with this feature representation, we can add any number of new features to the set $\mathbf{E}$ with ease as the feature sets corresponding to user and item are independent. A new feature could be another item or a user specific feature, for example user age, location, etc. Similarly, we can define feature set $\mathbf{F}$ as a set of users where $|F| = L = M = |\mathcal{U}|$. In otherwords, the 'features' that represent each item are associated with individual users – e.g., 'this is an example kind of user that the item is made for'. So each l^{th} feature in $\mathbf{F}$ corresponds to the l^{th} user in $\mathcal{U}$.

In summary, we defined u-features as a set of items and v-features as a set of users. As we defined $\mathbf{E}$ and $\mathbf{F}$, u-preference and v-appeal mappings are straightforward as they are simply mappings between elements of $\mathbf{E}$ and $\mathbf{F}$ feature sets (elite properties). That is, the u-preference is a function on $\{\mathbf{E} \times \mathbf{F}\}$ and the corresponding binary function values are represented by the matrix $\mathbf{G}$. Similarly, v-appeal is a function on $\{\mathbf{F} \times \mathbf{E}\}$ and the binary function values are represented by the matrix $\mathbf{H}$. In a collaborative filtering scenario, with close observation, these mappings are actually between user and item sets as the user features are the "set of items" and item features are the "set of users".

Now let's take a look at how the user preferences and item appeal are expressed in our unified information matching framework for collaborative filtering. In our framework, each individual user is assumed to have preferences for certain "kinds of items" similar to our example in chapter 3 where employer seeks a candidate with certain characteristics for a job vacancy. As we initially have no external indication of what "kinds" of items exist, this preference function is an unknown over the entire item space. That is, each item has a preference value for this user – not as an individual item, but as a representative of "items like this". In the model these preferences in model are represented through user elite properties and the forward preference matrix $\mathbf{G}$. In an exactly dual form, each individual item is assumed to have *appeal* to different kinds of users. Each user has an appeal value for this item – not as an individual user, but as a representative of "users like this one". Similarly, item preferences (*appeal*) as above are represented through item elite properties and the backward preference function/matrix $\mathbf{H}$. When an individual user sees an individual item, his/her reaction (rating, rt_{mn}) is assumed to be a stochastic function of the combination of user-item preference and item-user appeal representing the directional preferences from both sides (bi-directional preferences).

Finally, we need a set of relevant pairs for the retrieval model based on our framework in order to learn parameters for representing each user and item in u-features and v-features respectively and also to learn u-preferences and v-appeal for evaluating the bi-directional preferences. The simplest and most common approach to create a set of relevant pairs is to define a relevance threshold over the observed set of ratings between user, item pairs and to identify of a set of relevance observations (both relevant and non-relevant) [BCC11]. We can identify $\mathcal{D}^r \overset{def}{=} \{r_{mn}\}$, a data set of relevance observations, by defining the binary relevance between user-item, $\mathbf{u}_m, \mathbf{v}_n$ as follows:

$$r_{mn} = \begin{cases} 1, & \text{if } rt_{mn} > t \quad \text{where } \; t \text{ is a relevance threshold} \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

In our unified information matching framework, a *relevant* user-item pair satisfies the directional preferences of both the user and item. Therefore, as per our definition, every observed rating between a user-item pair is a result of a stochastic function of bi-directional preferences, and a result of stochastic function on a relevant pair generating a rating $> t$.

We have identified and defined u-feature, v-feature sets and u-preferences, v-appeal mappings and a set of relevant pairs. Now, as per the model, we can estimate the probability of relevance $p(r_{mn} = 1|\mathbf{u}_m, \mathbf{v}_n)$ using the $\mathbf{u}_m$ representation over the set of items (u-features), $\mathbf{v}_n$ representation over the set of users (v-features) and the mappings between users' features to items' features.

In order to estimate the probabilistic representations of user and item, in addition to relevance pairs, we can use their observed ratings on corresponding items, users as feature values for user and item respectively. That means, from the probabilistic model in chapter 3, $x_m^n \equiv rt_{mn}$ and $y_n^m \equiv rt_{mn}$. Note that we can use the same rating as an observed feature value when a user represented with a corresponding item as a feature and vice-versa. However, according to the model estimation formulation, the ratings received by the item from other users' and the user ratings on other items have an influence on how they describe each other when one of them is used as a feature and other as an object. In other-words, the user probabilistic representation with an item as a feature is not same as the probabilistic representation of the item with the same user as a feature because of the influence from other ratings of the user and item. Once we estimate the parameters of the model, we can compute the probability of relevance between any user-item pair by representing them in corresponding user and item feature spaces.

4.2.2 Discussion

In order to apply our matching framework to CF, all we needed was to define user, item feature spaces and a set of relevant pairs. Additionally, the feature space for both users and item is not required to be a common feature space. This shows its application in collaborative filtering where there are no common features/attributes between users and items. Another important aspect about this application is that the only information available to the model is the relevance information between the user-item pairs. The unified matching framework seamlessly solves the relevance matching in CF and at the same time it can also utilise any user specific or item specific features by simply adding those features to u-features and v-features respectively, and in principle the u-preference and v-appeal matrices can map between any sets of features.

There are a number of different advantages of using our unified information matching framework for CF. Firstly, with this framework, it is easy to see that the retrieval model does not require a common feature space to model information objects. It allows for the users and items to be modelled independent of one another (if necessary), and can incorporate any available information that is solely specific to users or items, and still compute relevance matching. This is unlike the matrix factorization methods and dimension reduction methods, such as SVD [KBV09] and topic models [Hof04] to name just a few, where we need to set any specific number of hidden common feature dimensions in which both the users and items will be represented.

Secondly, the model does not compute any explicit similarity between users or items in computing the relevance. So there is no need to explicitly compute the similarities between the users or items, which is the basis of the user-based approaches [SM95] and the item-based approaches [SKKR01]. This is different compared to the unified collaborative filtering model presented by Wang *et al.* in [WdVR08], where an unknown rating is estimated by *explicit similarity measures* from three sources: the user's own ratings for different items (item-based), other users ratings for the same item (user-based), and ratings from different but similar users for other but similar items.

Finally, even though the unified matching framework does not compute explicit similarities between users or items, it exploits the implicit similarity between users and items. To understand how it does this, let us have a close look at the preference, appeal matrices $\mathbf{G}$ and $\mathbf{H}$ respectively. Considering $\mathbf{G}$, insofar as it maps users into v-features based on u-features and implicitly identifies all the users with potentially the same or different u-features and maps to the same set of v-features. In this case a user might not start with the same preferences, but after learning based on the relevance the model will implicitly map all the similar users into the same set of item features. In document retrieval, this function of $\mathbf{G}$ would emerge in a relevance feedback environment, from different users identifying the same items as relevant to their needs. Similarly, the matrix $\mathbf{H}$ will identify similar items, by mapping them onto the same user properties. These characteristics of the matrices can only be expected to emerge in a *relevance feedback* environment; the model learns those implicit similarities whenever there is relevance data. In CF, all the available data is relevance data, so it naturally learns the implicit similarities.

In the following section, we use the our theory and framework to derive a relevance ranking function in a very similar way to how traditional relevance ranking functions are derived in IR, e.g. BM25 [RZ09]. The idea is to use the derived relevance ranking function to perform some experiments to validate the theory and framework.

4.2.3 Relevance Ranking Function

The objective of the ranking function in CF is to rank a set of items for a given user based on their probability of relevance. In this section, we derive the probabilistic relevance ranking function much like the traditional models, e.g., BM25 [RZ09]. That is, our objective is to compute $p(r = 1|\mathbf{u}_m, \mathbf{v}_n)$ where r is a binary relevance random variable. Now, before we derive the ranking function, let us assume that $\mathbf{E}^m, \mathbf{F}^n$ are the binary vector representations of $\mathbf{u}_m$ and $\mathbf{v}_n$ respectively over $\mathbf{E}$ and $\mathbf{F}$ elite property spaces respectively where $\mathbf{E}^m \in \{0,1\}^{|\mathbf{E}|}$ and $\mathbf{F}^n \in \{0,1\}^{|\mathbf{F}|}$.

Now, we compute the probability of relevance by marginalising over $\mathbf{E}^m, \mathbf{E}^n, \mathbf{G}$ and $\mathbf{H}$ in the following way:

$$p(r = 1|\mathbf{u}_m, \mathbf{v}_n) = \sum_{\mathbf{E}^m} \sum_{\mathbf{F}^n} \sum_{\mathbf{G}} \sum_{\mathbf{H}} p(r = 1, \mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}|\mathbf{u}_m, \mathbf{v}_n) \tag{4.6}$$

Eq. 4.6 is the result of marginalising over all possible combinations of binary vectors $\mathbf{E}^m, \mathbf{F}^n$ and matrices $\mathbf{G}, \mathbf{H}$.

By applying Bayesian transformation, we can rewrite Eq. (4.6) as,

$$p(r=1|\mathbf{u}_m, \mathbf{v}_n) = \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}} p(r=1|\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}, \mathbf{u}_m, \mathbf{v}_n) p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}|\mathbf{u}_m, \mathbf{v}_n) \quad (4.7)$$

From our theory and framework, we know that:

- The elite properties (features) of the $\mathbf{u}_m, \mathbf{v}_n$ ($\mathbf{E}^m, \mathbf{F}^n$) and their preference mappings in $\mathbf{G}, \mathbf{H}$ are sufficient to determine the relevance between $\mathbf{u}_m, \mathbf{v}_n$.
- $\mathbf{E}^m$ is only dependent on $\mathbf{u}_m$, $\mathbf{F}^n$ is only dependent on $\mathbf{v}_n$ and $\mathbf{G}, \mathbf{H}$ are independent of each other as well as $\mathbf{u}_m$ and $\mathbf{v}_n$.

By applying the above independent assumptions to Eq. (4.7) we get,

$$p(r=1|\mathbf{u}_m, \mathbf{v}_n) = \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}} \underbrace{p(r=1|\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H})}_{\text{part 1}} \,.\, p(\mathbf{E}^m|\mathbf{u}_m) \,.\, p(\mathbf{F}^n|\mathbf{v}_n) \,.\, p(\mathbf{G}) \,.\, p(\mathbf{H}) \quad (4.8)$$

Now, we further reduce the above probability relevance function in Eq. (4.8) by applying Bayesian transformation to part 1 of the equation. By applying the transformation we get,

$$p(r=1|\mathbf{u}_m, \mathbf{v}_n) = \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}} \frac{p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}|r=1)p(r=1)}{p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H})} \,.\, p(\mathbf{E}^m|\mathbf{u}_m) \,.\, p(\mathbf{F}^n|\mathbf{v}_n) \,.\, p(\mathbf{G}) \,.\, p(\mathbf{H})$$

In the above equation, we notice that $p(r = 1)$ is a constant and $p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}) = p(\mathbf{E}^m) \,.\, p(\mathbf{F}^n) \,.\, p(\mathbf{G}) \,.\, p(\mathbf{H})$ as they are independent. By taking the constant out of the summation and substituting the independence into the above equation we get,

$$p(r=1|\mathbf{u}_m, \mathbf{v}_n) = p(r=1) \,.\, \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}} \frac{p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}|r=1)}{p(\mathbf{E}^m) \,.\, p(\mathbf{F}^n)} \,.\, p(\mathbf{E}^m|\mathbf{u}_m) \,.\, p(\mathbf{F}^n|\mathbf{v}_n) \quad (4.9)$$

The constant $p(r = 1)$ will not alter the ranking order of $\mathbf{v}_n$'s for a given $\mathbf{u}_m$ if the ranking is based on the probability of relevance computed in Eq. (4.9). As we are interested only in ranking, we ignore the constant and rewrite the relevance ranking function as,

$$p(r=1|\mathbf{u}_m, \mathbf{v}_n) \propto_r \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}} p(\mathbf{E}^m, \mathbf{F}^n, \mathbf{G}, \mathbf{H}|r=1) \,.\, \frac{p(\mathbf{E}^m|\mathbf{u}_m)}{p(\mathbf{E}^m)} \,.\, \frac{p(\mathbf{F}^n|\mathbf{v}_n)}{p(\mathbf{F}^n)} \quad (4.10)$$

where $\propto_r$ is rank equivalence.

In our framework, we assumed that each user property's binary eliteness/description value (or binary feature value) for the corresponding $\mathbf{u}_m$ is independent of other properties, and item property's eliteness value for $\mathbf{v}_n$ is independent of other properties. Similarly, we also made an assumption that each binary mapping value in $\mathbf{G}$ is independent of other values in $\mathbf{G}$ and similarly for the value in $\mathbf{H}$.[1]

[1] A "user property" seeking an "item property" is independent of other properties and vice-versa.

Now, let α_k be the binary eliteness value of the k^{th} user elite property in vector $\mathbf{E}^m$ and β_l be binary eliteness value of the l^{th} item property in $\mathbf{F}^n$. Similarly, we denote by γ_k^l, δ_l^k the binary mapping values of $\mathbf{G}(k,l)$ and $\mathbf{H}(l,k)$ respectively.

Now, based on independence assumptions, we can write Eq.(4.10) as

$$p(r=1|\mathbf{u}_m,\mathbf{v}_n) \propto_r \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}}\prod_{k=1}^{N}\prod_{l=1}^{M} p(\alpha_k,\beta_l,\gamma_k^l,\delta_l^k|r=1)\ .\ \frac{p(\alpha_k|\mathbf{u}_m)}{p(\alpha_k)}\ .\ \frac{p(\beta_l|\mathbf{v}_n)}{p(\beta_l)} \tag{4.11}$$

The Eq.(4.11) is our final relevance ranking function. Before getting into further details on the ranking function, let $\Omega(m), \Pi(n)$ be the sets of observed ratings of $\mathbf{u}_m$ and $\mathbf{v}_n$ respectively. We note that we could have derived the relevance ranking function by estimating the joint probability of $p(r = 1, \Omega(m), \Pi(n)|\mathbf{u}_m, \mathbf{v}_n)$ instead of just $p(r = 1|\mathbf{u}_m, \mathbf{v}_n)$ and then use $\Omega(m)$ and $\Pi(n)$ for estimating the uncertainty in binary elite property values α_k, β_l for $\mathbf{u}_m$ and $\mathbf{v}_n$ respectively. The only reason for not doing so is to keep the derivation and ranking function more general. However, we will use $\Omega(m), \Pi(n)$ for estimating the eliteness of properties for $\mathbf{u}_m, \mathbf{v}_n$.

The final ranking function has three parts in its Eq. 4.11 as shown below,

- Joint probability of elite properties & their mappings

$$p(r=1|\mathbf{u}_m,\mathbf{v}_n) \propto_r \sum_{\mathbf{E}^m}\sum_{\mathbf{F}^n}\sum_{\mathbf{G}}\sum_{\mathbf{H}}\prod_{k=1}^{N}\prod_{l=1}^{M} \boxed{p(\alpha_k,\beta_l,\gamma_k^l,\delta_l^k|r=1)}\ .\ \boxed{\frac{p(\alpha_k|\mathbf{u}_m)}{p(\alpha_k)}}\ .\ \boxed{\frac{p(\beta_l|\mathbf{v}_n)}{p(\beta_l)}}$$

- $\mathbf{u}_m$'s elite property information
- $\mathbf{v}_n$'s elite property information

In the above ranking function, the computation of the summation over all possible binary vectors and matrices is expensive. Therefore, we make some simple but valid assumptions to reduce the computation and derive a much simpler ranking function.

1. We first create a binary relevance pairs data set, i.e., $\mathcal{D}^r$, by setting a threshold on the observed rating between the user-item pairs.

2. We set the $\mathbf{G}$ and $\mathbf{H}$ mapping functions based on the following observations:

   (a) The set of user properties/features are items and vice-versa.

   (b) If $r_{mn} = 1$ then the bi-directional preferences are satisfied between $\mathbf{u}_m$ and $\mathbf{v}_n$. This implies that if $\alpha_k = 1, \beta_l = 1$ for $\mathbf{u}_m$ and $\mathbf{v}_n$ respectively then $\gamma_n^m = 1$ ($\mathbf{G}(n,m) = 1$) and $\delta_m^n = 1(\mathbf{H}(m,n) = 1)$ (bi-directional condition).

   (c) From (b), by using $\mathcal{D}^r$, we set the γ_k^l and δ_l^k value as "1" for all the property mappings in $\mathbf{G}, \mathbf{H}$ corresponding to the relevant user-item pairs. That is, if $(\mathbf{u}_m, \mathbf{v}_n)$ is a relevant pair then $\mathbf{G}(n,m) = 1 = \mathbf{H}(m,n)$.

3. Using (2), we can set $p(\alpha_k, \beta_l, \gamma_k^l, \delta_l^k | r = 1) = 0$ if $(\gamma_k^l = 0 \vee \delta_l^k = 0)$ as the probability is conditioned on $r = 1$.

4. Now, let's take a look into the values of α_k, β_l in the relevant set of pairs ($r = 1$).

   (a) From the logical relationship in Eq. (4.4), bi-directional preference matching conditions, we know that

   $$r = 1 \Rightarrow z^{kl} = 1 \Rightarrow \Big(\big((1-\alpha_k)(1-\delta_l^k)\beta_l\big) \vee \big(\alpha_k(1-\beta_l)(1-\gamma_l^k)\big) \vee (\alpha_k\beta_l) \vee \big((1-\alpha_k)(1-\beta_l)\big) \Big)$$

   (b) But from (3) we know that $\gamma_k^l = \delta_l^k = 1$ if $r = 1$. By substituting the values, we can rewrite the above logical expression as follows,

   $$r = 1 \Rightarrow \Big((\alpha_k\beta_l) \vee \big((1-\alpha_k)(1-\beta_l)\big) \Big)$$

   The new expression implies that if $r = 1$ then $\alpha_k = \beta_l = 1$ or $\alpha_k = \beta_l = 0$.

5. By using $\alpha_k, \beta_l, \gamma_k^l, \delta_l^k$ values from (3) and (4) we can deduce the following:

$$p(\alpha_k, \beta_l, \gamma_k^l, \delta_l^k | r = 1) = \begin{cases} 1, & \text{if } \Big[\Big((\alpha_k = \beta_l = 0) \vee (\alpha_k = \beta_l = 1) \Big) \wedge \Big(\gamma_k^l = \delta_l^k = 1 \Big) \Big] \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

Now, we rewrite the relevance ranking function, by substituting $p(\alpha_k, \beta_l, \gamma_k^l, \delta_l^k | r = 1)$ value from (4.12) into Eq. (4.11), as follows:

$$p(r = 1|\mathbf{u}_m, \mathbf{v}_n) \propto_r \prod_{(\text{k, l}):r_{kl} \in \mathcal{D}^r} \Bigg[\underbrace{\frac{p(\alpha_k = 1|\mathbf{u}_m)}{p(\alpha_k = 1)} \cdot \frac{p(\beta_l = 1|\mathbf{v}_n)}{p(\beta_l = 1)}}_{\text{part 1}} + \underbrace{\frac{p(\alpha_k = 0|\mathbf{u}_m)}{p(\alpha_k = 0)} \cdot \frac{p(\beta_l = 0|\mathbf{v}_n)}{p(\beta_l = 0)}}_{\text{part 2}} \Bigg] \tag{4.13}$$

In general terms the ranking function in Eq. (4.13) can be described as follows:

- At the end of iterating over all the relevant pairs, the over all quantity computed in part 1 aggregates to a measure indicating how well $\mathbf{u}_m$ and $\mathbf{v}_n$ pair share the similar properties in comparison to a set of known relevant pairs. The more they share with relevant pairs the more relevant they are according the function. If we observe closely, the ranking function is computing this quantity by simply iterating over all the relevant pairs and computing how well user-item pair share similar properties to those of relevant pairs.

  It is using relevance information on $\mathbf{u}_m$, $\mathbf{v}_n$ and all other user-item pairs going through all the relevant pairs which include the past relevance information on $\mathbf{u}_m$ and $\mathbf{v}_n$.

- Similarly, part 2 computes the quantity that measures how well do $\mathbf{u}_m$ and $\mathbf{v}_n$ not share features that do not describe relevant pairs.

As a combination of part 1 and 2, the ranking function computes how likely it is that this $\mathbf{u}_m$ and $\mathbf{v}_n$ shares the properties of the relevant data set in two distinct ways and computes the relevance. The key point here is that the ranking function is simply using only relevance data in computing the probability of relevance and serves our objective of theoretical verification our framework in using relevance information on both users and items. The Eq. (4.13) is the final collaborative filtering ranking function.

Even though the ranking function is simple, we will see in chapter 5 that it outperforms the existing frameworks in recommending the top relevant items to a given user on different real world data sets.

4.2.4 Computing probabilities

To estimate the probabilities in Eq. (4.13), we make an assumption that the observed rating of a user-item pair, rt_{mn}, has a stochastic element associated with the item's appeal to the 'kind of user' $\mathbf{u}_m$ belongs to and the user's preference for the 'kind of item' that $\mathbf{v}_n$ belongs to. Now, in order to estimate the preference distribution of an individual user over 'kinds of items', we further assume that this user's observed ratings are the result *only* of this user's preferences. Similarly, to estimate the appeal distribution of an individual item over kinds of user, we assume that the observed ratings on this item are the result *only* of this item's appeal. These two assumptions are clearly oversimplifications but more sophisticated models can be pursued in future work.

Using the assumptions, we compute the probability that the kind of item $\mathbf{v}_k$ appeals to the user $\mathbf{u}_m$ as $p(\alpha_k = 1|\mathbf{u}_m) \equiv p(\alpha_k = 1|rt_{mk})$ where $rt_{mk} \in \mathcal{D}^{rt}$ denotes the observed user $\mathbf{u}_m$ rating on item $\mathbf{v}_k$. Similarly, we compute the probability that the kind of user $\mathbf{u}_l$ prefers the item $\mathbf{v}_n$ as $p(\beta_l = 1|\mathbf{v}_n) \equiv p(\beta_l = 1|rt_{ln})$.

We make another assumption that ratings given by a 'kind of user' $\mathbf{u}_l$ to a set of items follows one distribution in the 'kind of items' she/he prefers, and another distribution in non-preferred 'kind of items'. Similarly, the ratings received by a 'kind of item' $\mathbf{v}_k$ follows one distribution in the 'kind of users' that the item appeals to and another distribution in the ratings received from the 'kind of users' it does not appeal to. Now, we can do the probabilistic inference about the preference of a user from his associated ratings over a set of items.

By applying Bayes' rule to $p(\alpha_k = 1|rt_{mk})$, we get

$$p(\alpha_k = 1|rt_{mk}) = \frac{p(rt_{mk}|\alpha_k = 1)p(\alpha_k = 1)}{\sum_{\alpha_k \in \{0,1\}} p(rt_{mk}|\alpha_k)p(\alpha_k)}. \tag{4.14}$$

$$p(\beta_l = 1|rt_{nl}) = \frac{p(rt_{nl}|\beta_l = 1)p(\beta_l = 1)}{\sum_{\beta_l \in \{0,1\}} p(rt_{ln}|\beta_l)p(\beta_l)}. \tag{4.15}$$

Using the above values we compute $p(\alpha_k = 0|rt_{mk}) = 1 - p(\alpha_k = 1|rt_{mk})$ and $p(\beta_l = 0|rt_{nl}) = 1 - p(\beta_l = 1|rt_{nl})$. By substituting above expressions into Eq.(4.13), we get the ranking function,

$$\begin{aligned} &p(r = 1|\mathbf{u}_m, \mathbf{v}_n) \propto_r \qquad (4.16)\\ &\prod_{(\text{k, l}):r_{kl} \in \mathcal{D}^r} \left(\frac{p(rt_{mk}|\alpha_k = 1)}{\sum_{\alpha_k \in \{0,1\}} p(rt_{mk}|\alpha_k)p(\alpha_k)} \cdot \frac{p(rt_{nl}|\beta_l = 1)p(\beta_l = 1)}{\sum_{\beta_l \in \{0,1\}} p(rt_{ln}|\beta_l)} \right)\\ &\quad + \left(1 - \frac{p(rt_{mk}|\alpha_k = 1)p(\alpha_k = 1)}{\sum_{\alpha_k \in \{0,1\}} p(rt_{mk}|\alpha_k)p(\alpha_k)}\right) \cdot \left(1 - \frac{p(rt_{nl}|\beta_l = 1)p(\beta_l = 1)}{\sum_{\beta_l \in \{0,1\}} p(rt_{ln}|\beta_l)p(\beta_l)}\right) \end{aligned}$$

To estimate the probabilities in the above equation, we use a version of the 2-Poisson mixture used in ad-hoc retrieval [RW94]. We assume that the item's received ratings ($\Pi(k)$) from the "kinds of users" to which it appeals follow a Poisson distribution, and a different Poisson distribution among users to whom it does not appeal. We make a similar assumption about the ratings of a user. In collaborative filtering, modelling of ratings with Poisson has an advantage of scaling the models to massive data sets because of the mathematical form of Poisson likelihood [GHB13]. Moreover, the 2-Poisson modelling has been proven to be successful in ad-hoc text retrieval [RW94]. In the last section of this chapter, we present parameter estimation methods for 2-Poisson mixture and subsequently use the methods for conducting the experiments in Chapter 5.

4.3 Probabilistic Group Recommendation

The problem of group recommendation, as stated previously, is to recommend a set of items to a group of individuals by ranking items based on the estimated relevance. The fundamental objective of this application is to show that we can extend our theory and framework to incorporate extra contextual information into the relevance ranking model. Therefore, in the following, we describe how the above collaborative filtering model can be augmented in order to provide group recommendations. In order to derive a group relevance ranking function, we first define a group, describe different aspects of group recommendation and our notation, and provide a definition for item relevance to a group. Then, we describe a framework that uses our unified information matching, sets of users, and group ranking strategies to compute item relevance to groups.

A *group* is a set of individuals who are set in a shared context (e.g., members of the same household, friends, etc.) and who may have common interests. Each individual will have both *personal* and *group* preferences: interests that they are likely to pursue as individuals, which may be unique and independent from others, and preferences that arise from being a member of the group, which will thus be shared with other group members. The relevance of an item to any given group will vary with the type of group that seeks recommendations. There are many different types of such groups that could be defined. For example:

1. **Consensus Preference Group**. If a group is a set of friends who want to do something together (e.g., go to the cinema), then the recommended items should be relevant to all individuals, or at least not disliked by any one member in those instances where no consensus is available.

2. **Shared Preference Group**. For groups that, again, seek items to be enjoyed together, but with conditions that are more relaxed than consensus preference groups: recommended items should be relevant to all members, or at least not disliked by the majority when no consensus is available.

3. **Split Preference Group**. If a group consists of a household who wants recommendations as a unit, but may "consume" the items at different times (e.g., members of a family watching different television programmes), then the relevance of an item will be determined by whether there exists at least one user who likes the item.

Research to date in this domain has proposed two approaches: computing recommendations for the group by merging any members' ratings into a single profile, or computing ranked recommendations for each individual which are then merged via a range of heuristics. In doing so, none of the past approaches reason on the preferences that arise in individuals *when they are members of a group*. To understand the problem, consider the example where a family is sharing a single television. Each individual member of the family might want to see some programmes alone (e.g., late night shows). In this scenario, both the teenage boy and the elder member of the family might like to watch these programmes but not as a group. If we don't consider the preferences of a group, we might recommend the items they don't want to watch together. So, a group recommendation model has to consider both the preferences of individuals and of the group. And the group information acts as contextual information with the user preferences.

In the following, we use these notions of group relevance to extend the information matching probabilistic framework so that it caters for recommending items to groups. Before we develop a group relevance ranking function we define our relevance hypothesis as follows,

Group Relevance Hypothesis (GRH) *The relevance between a group and an item* $\mathbf{v}_n$ *is only dependent on the relevance of* $\mathbf{v}_n$ *to individual members of the group.*

Using this hypothesis, we derive a probabilistic group recommendation framework that not only includes the preferences of individual users' but also integrates the users preferences when they are in a group while recommending a set of items.

First, we define the following notation:

1. We denote by $\mathcal{G}$ a group containing a set of h users; Υ is the set of users $\Upsilon = \{\mathbf{u}_1, \mathbf{u}_2 \cdots, \mathbf{u}_h\}$.
2. r_g is the binary *group* relevance between any group-item pair, where
$$r_g = \begin{cases} 1, & \text{indicate relevance} \\ 0, & \text{indicate non relevance} \end{cases} \tag{4.17}$$
3. $\Re$ is a vector of r_{hn} values containing the relevance of each user $\mathbf{u}_h \in \Upsilon$ to a given item $\mathbf{v}_n$.

In order to recommend a set of items to a group $\mathcal{G}$, we therefore need to calculate $p(r_g = 1|\mathcal{G}, \mathbf{v}_n)$ for each item $\mathbf{v}_n$, or the probability of relevance between the group $\mathcal{G}$ and each individual item $\mathbf{v}_n$ in the collection. We derive this probability as follows. First, the probability of an item's relevance to a group is defined as:
$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) = \sum_{\Re} p(r_g = 1, \Upsilon, \Re|\mathcal{G}, \mathbf{v}_n) \tag{4.18}$$
By expanding $\mathcal{G}$ to its constituent users, we obtain:
$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) = \sum_{\Re} \underbrace{p(r_g = 1|\Upsilon, \Re, \mathbf{v}_n, \mathcal{G})}_{\text{Part 1}} \cdot \underbrace{p(\Upsilon, \Re|\mathcal{G}, \mathbf{v}_n)}_{\text{Part 2}} \tag{4.19}$$
From the **GRH** we know that the r_g is dependent only on $\Upsilon, \Re, \mathbf{v}_n$, so we can ignore $\mathcal{G}$ in part 1. Similarly, we can ignore $\mathcal{G}$ in part 2, since group relevance of an item is dependent only on the relevance of each individual member of the group to that item (as per our hypothesis). The Bayesian transformation then allows us to then rewrite Eq. (4.19) above as:
$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) = p(r_g = 1) \cdot \sum_{\Re} p(\Re, \mathbf{u}_1, \cdots, \mathbf{u}_h, \mathbf{v}_n|r_g = 1) \cdot p(\Re|\mathbf{u}_1, \cdots, \mathbf{u}_h, \mathbf{v}_n) \tag{4.20}$$

Assuming that the relevance $r_j \in \Re$ is dependent only on $\mathbf{u}_j, \mathbf{v}_n$ and eliminating the constant $p(r_g = 1)$, we can rewrite the above equation as:

$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) \propto_{r_g} \sum_{\Re} \underbrace{\left\{ \prod_{j=1}^{h} p(r_j, \mathbf{u}_j, i | r_g = 1) \right\}}_{\text{Part 1}} \underbrace{\left\{ \prod_{j=1}^{h} p(r_j | \mathbf{u}_j, \mathbf{v}_n) \right\}}_{\text{Part 2}} \tag{4.21}$$

- Group relevance

$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) \propto_{r_g} \sum_{\Re} \prod_{j=1}^{h} \boxed{p(r_j, \mathbf{u}_j, i | r_g = 1)} \cdot \boxed{p(r_j | \mathbf{u}_j, \mathbf{v}_n)}$$

- Individual relevance

Using Eq. (4.21), we obtain a probabilistic model for group recommendation that accounts for both group (Part 1) and user (Part 2) relevance. The *Group* part of the Eq. (4.21) captures the $\mathbf{v}_n$ relevance to $\mathbf{u}_j$ when she/he is in a group whereas the *Individual* captures the relevance of $\mathbf{v}_n$ to $\mathbf{u}_j$ as an individual. We can compute the relevance of $\mathbf{v}_n$ to each individual user $\mathbf{u}_h$ using the unified information matching framework, Eq. (4.13).

4.3.1 Discussion

The group probabilistic relevance ranking function in Eq.(4.21) combines the individual preferences of a member along with his/her behavior in a group. Therefore, if the retrieval system has access to members' information in different groups, it can be seamlessly integrated into our group relevance matching framework. Another interesting aspect of the framework is that we can incorporate various scenarios of group matching into relevance ranking. We show some of the strategies in the following section. One key attribute that allows us to model groups and individuals together is the fact that the probability of relevance between users, items with group preferences are computed in a single probability space allowing us to compare the relevance between two user-item pairs. In other words, a crucial aspect of the computation of relevance in Eq. (4.21) is that the probabilities of relevance between all the users in the group to all the items in the collection must be comparable, i.e. the probability space in which the probability of relevance between all user-item pairs is computed must be the same [Rob05, Rob03]. This is uniquely achievable by using the principles of (unified) information matching: most of the traditional relevance ranking functions used in information retrieval are not capable of estimating this relevance because they function on probability event spaces that are either conditioned on users (queries) [RZ09] or items (documents) [Zha08].

4.3.2 Example Group recommendation strategies

Assuming that the group recommendation scenario is to find items that all members like (i.e., the context is a consensus preference group), we know that for $r_g = 1$ then $r_j = 1$ for each $(\mathbf{u}_j, \mathbf{v}_n)$ where $\mathbf{u}_j \in \mathcal{G}$.

By turning this implication around, we set that $p(r_j = 1, \mathbf{u}_j, \mathbf{v}_n | r_g = 1) = 1$

Algorithm 1 Separately Calculate User Preference and Item Appeal

```
Compute each user description vector
for $\mathbf{u}_m \in Users$ U do
  for $k = 1 \cdots L$ do
    $\mathbf{E}_m[k] = \frac{p(\alpha_k = 1|\mathbf{u}_m)}{p(\alpha_k = 1)}$
  end for
end for
Compute each item description vector
for $\mathbf{v}_n \in Items$ V do
  for $l = 1 \cdots K$ do
    $\mathbf{F}_n[l] = \frac{p(\beta_l = 1|\mathbf{v}_n)}{p(\beta_l = 1)}$
  end for
end for
```

Algorithm 2 Given group $\mathcal{G}$, calculate the relevance $\hat{r}$ for each item for the group by Least Misery

```
$t =$ relevance threshold and $\mathcal{D}^r = (\mathbf{u}_e, \mathbf{v}_f) \in$ U $\times$ V where $r_{ef} \geq t$
for $\mathbf{u}_m \in \mathcal{G}$ do
  for unrated $\mathbf{v}_n \in Items$ V do
    $\hat{r}_{mn} = 1, \hat{r}(G, \mathbf{v}_n) = \infty$
    for $(m, n) \in \mathcal{D}^r$ do
      $\hat{r}_{mn} = \hat{r}_{mn} \times (\mathbf{E}_m[k] \times \mathbf{F}_n[l])$
    end for
    if $\hat{r}(\mathcal{G}, \mathbf{v}_n) \geq \hat{r}_{mn}$ then
      $\hat{r}(\mathcal{G}, \mathbf{v}_n) = \hat{r}_{mn}$
    end if
  end for
end for
```

and $p(r_j = 0, \mathbf{u}_j, \mathbf{v}_n | R_g = 1) = 0$. Finally, by substituting these into Eq. (4.21) we get:

$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) \propto_{r_g} \prod_{j=1}^{h} p(r_j = 1|\mathbf{u}_j, \mathbf{v}_n) \tag{4.22}$$

The Eq. (4.22) translates to saying that the probability of $\mathbf{v}_n$ being relevant to the group $\mathcal{G}$ is proportional to the individual satisfaction of each member in the group by considering all member's preferences are equally significant. We note that by defining the problem space probabilistically, any mechanism to estimate the relevance between a user-item can be substituted into the framework. For example, if the context contains split preference groups, items could be ranked with:

$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) \propto_{r_g} \min\left\{p(r_1 = 1|u_1, \mathbf{v}_n), \cdots, p(r_h = 1|\mathbf{u}_h, \mathbf{v}_n)\right\} \tag{4.23}$$

This captures the concept of *Least Misery* [BMR10] between the group members.

Least Misery strategy: In this strategy, the relevance score of an item to a group is the least relevance score of all the member's relevance in the group. The principal idea behind the strategy is that the item relevance to a group is the least happiest member's relevance. Therefor, if we rank the items based on the least misery score then the top items would give least misery to the entire group. To implement this strategy, we need to compute the relevance of the item to each individual user in the group and use the least relevance score as the group relevance for ranking.

In summary, we outlined the group recommendation algorithm that we have proposed here with the pseudocode in Algorithm 1 and 2. In Chapter 5, we test the performance of ranking function in Eq. (4.23) on a publicly available dataset.

4.4 Ad-hoc Text Retrieval

In previous sections, using the collaborative filtering (CF) problem, we have shown how the *information matching framework* (IMF) simultaneously utilises the relevance information available on all the information objects in computing the probability of relevance. In the CF problem, there were no common features between the user and the item information objects and the only information available to the system for computing the probability of relevance between user-item pairs was a set of relevance judgements in the form of ratings. In addition to relevance pairs, we have defined two distinct set of feature spaces and represented users and items in their respective feature spaces. The IMF combined the user and the item specific features information along with the available relevance information for computing the probability of relevance. We have also seen how easily we can add information other than relevance in computing the probability of relevance. Therefore, the case for utilising all the available relevance information on information objects along with non-common features information is not required. However, it is important to show the use of the *information matching framework* in computing probability of relevance when there is no relevance information available on information objects and the only available information to the retrieval system is on a common/shared set of features/properties between the information objects, e.g. vocabulary space between documents and queries. It is required partly because our framework relies on relevance information for learning preference mappings and updating information object representations. Therefore, our primary objective in this section is to show how the *information matching framework* can be applied in this scenario and show how we can estimate required preference mappings in the absence of objects interaction information (relevance).

In order to achieve our objective, we choose the ad-hoc text retrieval task as an application. The fundamental reason for choosing ad-hoc retrieval is the fact that the only information available to the retrieval system in computing the probability of relevance between a given document-query is the vocabulary terms and their statistics in queries and documents. Additionally, sometimes the system may have access to few relevance judgements on document-query pairs. However, when the relevance information is present, as seen before, it is rather straightforward to apply IMF for computing the probability of relevance.

The objective of the ad-hoc retrieval task is to rank a set of information items (documents) for a given information need (query) based on their probability of relevance. In order to apply IMF, we need to (a) identify two distinct feature spaces for representing the documents and queries respectively, (b) define mappings between the document and query feature spaces, and (c) gather a set of relevant document-query pairs for learning document, query and mapping representations.

Before we define features, mappings and derive the relevance ranking function, let's take a look at how the IMF can be presented as a solution framework for a document retrieval problem. An explanation of the IMF in terms of document retrieval would be as follows: an information need (represented by a query) with certain properties seeks an information item (document) with certain properties, and an information item with certain properties seeks to satisfy information needs with certain properties. These are the directional preferences of the item and the need. For example, if a query comes with an identified geolocation, this may (depending on the rest of the query) seek a document or page with a nearby geolocation (where the meaning of 'nearby' also depends on the rest of the query). Similarly, a page describing a restaurant will probably be 'seeking' relatively local people. On the other hand, we might hypothesise that any query is likely to seek an authoritative document (as measured by, say, PageRank). In the framework, these preferences are represented with the item and need features through the preference mappings (a-preference and b-preference as described in chapter 3).

The basic idea here is that the information need and information item are described with their respective features, potentially from different sets (we could think of these as *vocabularies*, but in principle the vocabulary for need-description is different from that for item-description). The matching for relevance then requires verifying *two separate directional preference* between these vocabularies: one from the need properties to the item properties (identifying which item properties are sought by each need), and one in the reverse direction (identifying which need properties are 'sought' by each item).

4.4.1 Features (Properties), Mappings and Relevance

Following the previous notation, let need, item features sets be $\mathbf{E}$ and $\mathbf{F}$ respectively. The need features ($\mathbf{E}$) could be a set of features associated with vocabulary words, query-specific features such as geolocation, query length, etc. Similarly, $\mathbf{F}$ could be a set of vocabulary term features, document-specific features such as PageRank, url depth, etc. However, in a traditional ad-hoc retrieval task, the only available information is vocabulary terms and their statistics in queries and documents. Therefore, in order to derive a simple ad-hoc retrieval ranking function, we define both $\mathbf{E}$ and $\mathbf{F}$ as a set of "k" features each corresponding to a single vocabulary term. We call this set of features "term-description" features and represent with Φ. We denote by $\mathbf{E}^m, \mathbf{F}^n$ be the binary vectors over the feature space Φ that generated $\mathbf{q}_m$ and $\mathbf{d}_n$ respectively as per IMF where $\mathbf{E}^m = \{e_1, e_2, \cdots, e_K\}$ and $\mathbf{F}^n = \{f_1, f_2, \cdot, f_L\}$ and e_k, f_l are the components of corresponding binary vectors.

If we have the relevance information, the framework learns the preference mappings between elements of set $\mathbf{E}$ to $\mathbf{F}$ and vice-versa based on the features associated with the relevance document-query pairs as illustrated previously in the collaborative filtering scenario. However, in traditional ad-hoc retrieval task with no relevance information it is not possible to learn the preference mapping matrices ($\mathbf{G}$ and $\mathbf{H}$). This problem can be circumvented by defining the preference mappings as an identity mapping between corresponding k^{th} features in $\mathbf{E}$ and $\mathbf{F}$, as k^{th} feature in each feature set is corresponding to the same “term-description” feature in the vocabulary.

Now, from the above definition, $\mathbf{G}$, F are mapping functions over Φ and defined as follows:

$$\mathbf{G} = \mathbf{H} = \begin{array}{c|cccc} & s_1 & s_2 & \cdots & s_k \\ \hline s_1 & 1 & 0 & \cdots & 0 \\ s_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_k & 0 & 0 & \cdots & 1 \end{array}$$

where $\gamma_k^l = \delta_l^k = 1$ if $k = l$, “0” otherwise.

Following the above relation, the definition of relevance between an information item (document) $\mathbf{d}_n$ and an information need (query) $\mathbf{q}_m$ under the verification of bi-directional preferences reduces to a simple relationship where $\mathbf{d}_n$ and $\mathbf{q}_m$ are relevant if and only if $\mathbf{E}^m = \mathbf{F}^n$, i.e. the binary feature value of all the features of $\mathbf{d}_n$ must be same as that of $\mathbf{q}_m$. We refer to this relevance relationship as relevance under “strict identity” relation. The reason for this reduction is that we do not need $\mathbf{G}$ and $\mathbf{H}$ for the computation of relevance as we know that the same features in Φ should describe the document and the query if they are relevant. In other words, the binary feature values corresponding to each feature in Φ must be same for both $\mathbf{d}_n$ and $\mathbf{q}_m$.

Now let’s derive a relevance ranking function for the ad-hoc retrieval task under the above definition of relevance. Before the derivation, we would like to note that the above approximated model is neither identical nor related to the matrix framework based model for information retrieval proposed by Rölleke *et al.* [RTK06] where the documents and queries are represented in matrix spaces and retrieval is performed a function of matrix operations applied to these spaces. However, it will be an interesting case study to look at the IMF in terms of the matrix operations as proposed by Rölleke *et al.* [RTK06].

4.4.2 Relevance Ranking Function

As per the definition of *relevance* in the previous section, the probability of relevance between $\mathbf{d}_n$ and $\mathbf{q}_m$, $p(r = 1|\mathbf{d}_n, \mathbf{q}_m)$, can be computed as

$$p(r = 1|\mathbf{d}_n, \mathbf{q}_m) \propto_r \sum_{\mathbf{E}^m} \sum_{\mathbf{F}^n} \prod_{i}^{|\Phi|} \underbrace{p(e_i = \alpha_i^m, f_i = \alpha_i^n | r = 1)}_{\text{part 1}} \cdot \frac{p(f_i = \alpha_i^n|\mathbf{d}_n)}{p(f_i = \alpha_i^n)} \cdot \frac{p(e_i = \alpha_i^m|\mathbf{q}_m)}{p(e_i = \alpha_i^m)} \tag{4.24}$$

where $\propto_r$ is rank equivalence (constant $p(r = 1)$ is ignored), $i \in \{1, 2 \cdots, |\Phi|\}$, α_i^m, α_i^n are the binary feature values of the i^{th} feature for the query and document respectively and e_i, f_i are i^{th} component of $\mathbf{E}^m$ and $\mathbf{F}^n$ respectively. The derivation of the above equation is exactly the same as the previous derivation used in collaborative filtering except that the feature set for query and document is the same in this case.

Eq. (4.24) is a relevance ranking function for ad-hoc retrieval when the same set of features can describe a document or query. Eq. (4.24) uses the information about the binary value of each feature for the given document and query ($p(f_i = \alpha_i^n|\mathbf{d}_n)$, $p(e_i = \alpha_i^m|\mathbf{q}_m)$), its value in the collection ($p(e_i)$, $p(f_i)$) and the joint probability of corresponding document-query feature values in relevant document-query pairs. If there is a new relevant pair, its information will be added when computing the relevance through part 1 in Eq. (4.24). Note that Eq. (4.24) corresponds to the relevance ranking function before applying our "strict identity" relevance condition.

In traditional TREC collections there is very little text on the query side. Therefore, in order to implement the ranking function we avoid the estimation of the probability of each feature in Φ taking a particular binary value for the given query by making the following assumption. *Query property description assumption:* As we have very little information (only two to three query terms) to infer the probability of query features taking a particular binary value, and the fact that each query term is very important in finding the relevant documents, we assume that each feature value corresponding to the query terms is "1" for the given query and others' feature values are "0", i.e. we know the binary vector $\mathbf{E}^m$. Basically, this assumption is similar to an implicit assumption that the query terms are elite to the query and other terms are non-elite as in [RW94]. In what follows, we use the terms *elite* and *non-elite* as synonymous with describing that a feature 'has a binary feature value 1' and 'has the binary feature value 0' respectively, for either users or items. For example, s_i is *elite* for $\mathbf{q}_m$ means that the binary feature value of the feature s_i for $\mathbf{q}_m$ is "1", i.e. $\alpha_i^m = 1$

Now, from the above assumption $e_i = 1$ (i^{th} component of $\mathbf{E}^m$) if $\mathbf{s}_i$ is elite for $\mathbf{q}_m$ otherwise $e_i = 0$. By applying Bayes' rule to the part 1 of Eq.(4.24) and factorising, we get

$$p(r=1|\mathbf{d}_n,\mathbf{q}_m) \propto_r \sum_{\mathbf{F}^n} \prod_{\forall i:e_i=0} \frac{P(f_i=\alpha_i^n|e_i=0,r=1)}{p(f_i=\alpha_i^n)p(e_i=0)} p(e_i=0|r=1)P(f_i=\alpha_i^n|\mathbf{d}_n)P(F_i=0|q)$$
$$\prod_{\forall i:e_i=1} \frac{p(f_i=\alpha_i^n|e_i=1,R=1)p(e_i=1|r=1)p(f_i=\alpha_i^n|\mathbf{d}_n)p(e_i=1|\mathbf{q}_m)}{p(f_i=\alpha_i^n)p(e_i=1)}$$

From the *Query property description assumption*, we know the value of each element in $\mathbf{E}^m$. So, we have $p(e_i = 1|\mathbf{q}_m) = 1$ if the term associated with the i^{th} property in Φ is present in query $\mathbf{q}_m$ and $p(e_i = 0|\mathbf{q}_m) = 0$ otherwise. By substituting these values, the above equation becomes

$$p(r=1|\mathbf{d}_n,\mathbf{q}_m) \propto_r \sum_{\alpha} \prod_{\forall i:e_i=0} \Big(\frac{p(f_i=\alpha_i^n|e_i=0,r=1)P(e_i=0|r=1)}{p(f_i=\alpha_i^n)p(e_i=0)} p(f_i=\alpha_i^n|\mathbf{d}_n)\Big)$$
$$\prod_{\forall i:e_i=1} \Big(\frac{p(f_i=\alpha_i^n|e_i=1,r=1)p(e_i=1|r=1)}{P(f_i=\alpha_i^n)P(e_i=1)} p(f_i=\alpha_i^n|\mathbf{d}_n)\Big)$$

As defined, a document and a query are relevant if and only if $\mathbf{E}^m = \mathbf{F}^n$. From the definition, if we know that $e_i = 1$ then the probability that of $f_i = 1$ ($\alpha_i^n = 1$) in the relevant set of documents is "1", i.e. $p(f_i = 1|r = 1, e_i = 1) = 1$, as they have the same value in the relevant set. Equally, it is the same when $e_i = 0$, i.e. $p(f_i = 0|r = 1, e_i = 0) = 1$. Note that from this assumption, the score of any vector in $\mathbf{F}^m$ is zero if $f_i = 0$ and $e_i = 1$ (or) $f_i = 1$ and $e_i = 0$ for at least one i. By substituting these values back into the previous equation, we get

$$p(r=1|\mathbf{d}_n,\mathbf{q}_m) \propto_r \prod_{\forall i:e_i=0} \frac{p(f_i=0|r=1)}{p(f_i=0)p(e_i=0)} p(f_i=0|\mathbf{d}_n) \prod_{\forall i:e_i} \frac{P(e_i=1|r=1)}{p(f_i=1)p(e_i=1)} p(f_i=1|\mathbf{d}_n) \tag{4.25}$$

Now, we can remove $p(e_i = 0|r = 1)$, $p(e_i = 1)$, $p(e_i = 0)$ and $p(e_i = 1|r = 1)$ in Eq. (4.25) as these

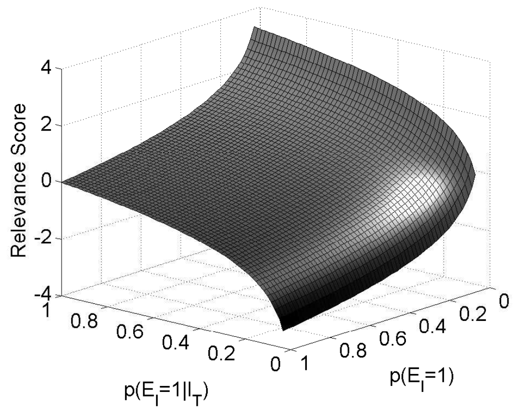


Figure 4.1: The behaviour of the probability description of a single feature.

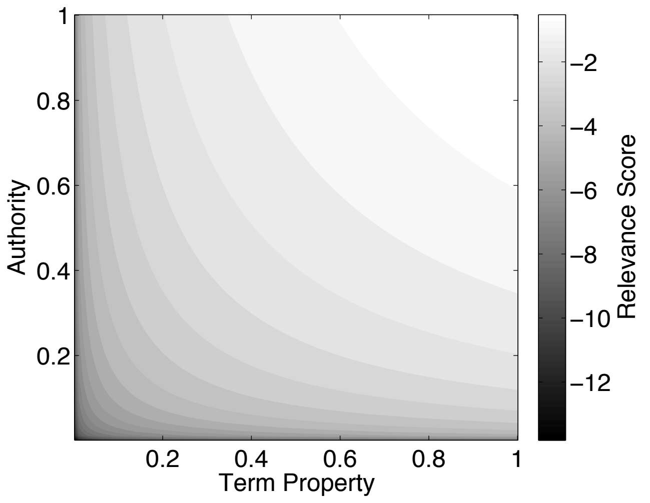


Figure 4.2: The behaviour of the relevance ranking score with two features.

.

Figure 4.3: The behaviour of relevance ranking function in information matching framework.

terms do not affect the ranking order. We thus get

$$p(r=1|\mathbf{d}_n,\mathbf{q}_m) \propto_r \prod_{\forall i:e_i=0} \frac{p(f_i=0|\mathbf{d}_n)}{p(f_i=0)} \cdot \prod_{\forall i:e_i=1} \frac{p(f_i=1|\mathbf{d}_n)}{p(f_i=1)} \tag{4.26}$$

Eq. (4.26) is a ranking function under the strict identity relation with the *Query property description assumption*. We ignore the terms with feature values "0" in Eq. (4.26) by assuming that the absence of terms without any relevance information represents unknown feature values. Then, applying a logarithm transform to the ranking function results in the following ranking function:

$$p(r=1|\mathbf{d}_n,\mathbf{q}_m) \propto_r \sum_{\forall i:e_i=1} \underbrace{\log \frac{p(f_i=1|\mathbf{d}_n)}{p(f_i=1)}}_{\text{single feature score.}} \tag{4.27}$$

The above simplification is similar to ignoring the terms that are not present in the query in [RSJ76, PC98, LZ01].

To see how the of scores each component in Eq. (4.27) affect the rankings score, we refer to the graph in Figure 4.3. The two base axes are the numerator of the property (the probability that the given feature describes the information items), and the denominator (the probability that the feature describes any information item in general). The vertical axis is the relevance score as the logarithm of the "single feature score" of Eq. (4.27). The maximum relevance score is achieved when $p(f_l = 1)$ is minimal and $p(f_l = 1|\mathbf{d}_m)$ (represented with information item I_T) is maximal as shown in Figure 4.1. This then implies that the feature describes very few information items (low $p(f_l = 1)$) but well describes the particular information item $\mathbf{d}_n$ ($p(f_l = 1|\mathbf{d}_m)$ is maximal). This is what one would expect of a reasonable ranking function. The Figure 4.2 shows how the relevance score changes when there are two properties (let's say, authority and a term property) where the X, Y axises describe each "Single feature score" and the Z axis is the sum of logarithmic scores of the two features. So, the overall relevance score depends on how important the features are in describing all the information items and how well each of them describes the information item (or need).

4.4.3 Interpretation of Inverse Document Frequency (IDF)

One of the by-products of our model is that the formula in Eq. (4.27) provides a yet another theoretical justification of IDF (inverse document frequency) as a scoring function [Jon72]. To see this, let us assume that the feature value of a feature corresponding to a term is "1" to a document if the term is present in the document and "0" otherwise. Then, the probability of the feature value being "1" in the collection is, $p(f_i = 1) \equiv \frac{n_{ci}}{N}$, where n_{ci} is the number of documents in the collection with the term associated with the i^{th} term-description feature/property, and N is the total number of documents in the collection. From the above assumption $p(f_i = 1|\mathbf{d}_n) = 1$ if $e_i = 1$ (term i is in the query). By substituting them in the ranking function in Eq. (4.27), we get

$$p(r = 1|\mathbf{d}_n, \mathbf{q}_m) \propto_r \sum_{\forall i:e_i} \log \frac{1}{p(f_i = 1)} = \sum_{\forall i:e_i=1} \log \frac{N}{n_{ci}} \tag{4.28}$$

Now, the ranking function in Eq. (4.28) is simply a function of the IDF values of the query terms. Essentially, it implies that the IDF score function relies on the assumption that a term is elite if it occurs in the document. This is different from the explanation provided by the Robertson-Spärck Jones model, where an explicit assumption that the whole collection is a non-relevant set is needed [RSJ76].

4.4.4 Computing Probabilities

To implement and test the ranking function in Eq. (4.27), we need to estimate the probabilities $p(f_i = 1|\mathbf{d}_n)$ and $p(f_i = 1)$ for each i^{th} property in Φ. In order to estimate the probabilities, we assume the following generative process where an author (or a user) will carry out the following process to express their information:

1. A user or author will choose a set of elite properties/features such that these properties/features can describe every aspect of the information that they want to express.

2. Once the features are chosen, an observable information item or need is generated by a stochastic function of the chosen features. The uncertainty about the description of the feature for the information item is injected during this generation process.

Now, we know that a document is generated from a set of term-description features. So, the occurrence of a term in a document has a stochastic element associated with the description of its corresponding term-description feature. Therefore, we compute the probability of the i^{th} term-description feature value being "1" for the given document $\mathbf{d}_n$ as $p(f_i = 1|\mathbf{d}_n) \equiv p(f_i = 1|tf_i)$ where tf_i denotes the term frequency associated with the i^{th} term-description feature in document $\mathbf{d}_n$. As we assume that the description of a feature (feature value) for a document is binary, from the hypothesis, a feature description is "1" for some documents in the collection and "0" for others. And, tf_i follows one distribution in a set of documents that were described by the feature and another distribution in a second set of documents that were not described by the feature. Therefore, we can draw a probabilistic inference about the description of a term-description feature from the frequency of its associated term in the document.

By applying Bayes' rule to $p(f_i = 1|tf_i)$, we get

$$p(f_i = 1|tf_i) = \frac{p(tf_i|f_i = 1)p(f_i = 1)}{\sum_{f_i \in \{0,1\}} p(tf_i|f_i)p(f_i)} \tag{4.29}$$

For simplicity, we use query terms to represent features that describe the query ($f_i = 1$ when $q_i = 1$). By substituting Eq. (4.29) back in Eq. (4.27), we get the following ranking function

$$p(r = 1|\mathbf{d}_n, \mathbf{q}_m) \propto_r \sum_{\forall i: e_i = 1} \log \frac{p(tf_i|f_i = 1)}{\sum_{f_i \in \{0,1\}} p(tf_i|f_i)p(f_i)} \tag{4.30}$$

In order to estimate the probabilities in Eq. (4.30), we assume that the collection of documents is a two component mixture for any given feature. As Φ is a set of term-description features, we assume that the term frequency of the term associated with each property in Φ follows a Poisson distribution in one set of documents that are described by the feature, $p(tf_i|f_i = 1) \equiv e^{-\mu_i(1)}\mu_i(1)^{tf_i}$, and another Poisson distribution in the other set, $p(tf_i|f_i = 0) \equiv e^{-\mu_i(0)}\mu_i(0)^{tf_i}$, where $\mu_i(1)$ and $\mu_i(0)$ are the two Poisson means. The mixing probability $p(f_i = 1) \equiv p_i$ is an additional parameter. This is the classic 2-Poisson mixture model [Har75a, RvRP80] with parameters $\mu_i(1), \mu_i(0), p_i$.

For inference in the above mixture model, we can approach either in a maximum likelihood (ML) or in a Bayesian framework coupled with Markov Chain Monte Carlo (MCMC) technique. For the experiments in the following chapter, we present a method for estimating the optimal parameter values of the mixture by using maximum likelihood estimation (using the Expectation Maximization (EM) algorithm [DLR77]) as well as Gibbs sampling for finite mixtures via MCMC [DR94]. By substituting the estimated parameter values in Eq.(4.27), we get the final ranking function

$$p(r = 1|\mathbf{d}_n, \mathbf{q}_m) \propto_r \sum_{\forall i: e_i = 1} log\Big(\frac{e^{-\mu_i(1)}\mu_i(1)^{tf_i}}{p_i e^{-\mu_i(1)}\mu_i(1)^{tf_i} + (1 - p_i)e^{-\mu_i(0)}\mu_i(0)^{tf_i}}\Big) \tag{4.31}$$

The ranking function in Eq.(4.31) looks similar to the ranking functions in [RvRP80, RW94] but is substantially different; the apparent similarity arises only from the use of the 2-Poisson distributional assumption. In chapter 5 we conduct experiments on some of the TREC collections using the ranking function (4.31).

Parameters	Description	Prior distribution
α_k^m and β_l^n	The uncertainty in $\mathcal{O}_m$, $\mathcal{O}_n$ being described/represented by the k^{th} feature in $\mathbf{E}$ and l^{th} feature in $\mathbf{F}$ respectively.	Beta
γ_k^l and δ_l^k	The uncertainty in mapping between the k^{th} and the l^{th} features $\big(\mathbf{G}(k,l)$ and $\mathbf{F}(l,k)\big)$.	Beta
λ_0^k, λ_1^k, λ_0^l and λ_1^l	The 2-Poisson mixture means corresponding to the k^{th} and the l^{th} features.	Gamma

Table 4.2: IMF model parameters and their conjugate prior distributions corresponding to the k^{th} feature in $\mathbf{E}$ and the l^{th} feature in $\mathbf{F}$, and information objects $\mathcal{O}_m \in \mathcal{A}$ and $\mathcal{O}_n \in \mathcal{B}$.

Parameters	Description
z_{mn}^{kl}, ω_k^m and ω_l^n	The z_{mn}^{kl} is a latent variable and represents the uncertainty in objects, $\mathcal{O}_m$ and $\mathcal{O}_n$, satisfying their bi-directional preferences with respect to the k^{th} and the l^{th} features. The ω's are data augmentation latent variables used introduced for optimisation A.

Table 4.3: IMF model latent parameters corresponding to k^{th} feature in $\mathbf{E}$ and l^{th} feature in $\mathbf{F}$, and information objects $\mathcal{O}_m \in \mathcal{A}$ and $\mathcal{O}_n \in \mathcal{B}$.

4.5 Parameter Estimation

In the previous sections, we have shown how the *information matching framework* can be used for three different retrieval scenarios where three different types of information are available to the retrieval system. In the following, we present a Bayesian inference method based on Variational approximate inference to infer the parameters in the graphical model for the IMF presented in section 3.5.1 of Chapter 3. Additionally, we present parameter estimation methods for estimating the parameters used in both collaborative filtering and ad hoc retrieval ranking functions in section 4.5.1.

4.5.1 Inference in the Bi-directional Unified Model (IMF model)

In section 3.5.1, we have presented a graphical model for the bi-directional unified model. In this section, we explain an optimisation procedure to estimate the model parameters based on variational inference and demonstrate the parameter estimation using an example. Previously, we defined the joint density by defining the set of parameters $\mathbf{\Theta}$, hyper parameters $\mathcal{H}$ and the observed data $\mathcal{D}$. Now, the objective is to estimate the parameters in $\mathbf{\Theta}$. In order to estimate the parameters we use Bayes' theorem to infer the posterior density of $\mathbf{\Theta}$ given $\mathcal{D}$ and $\mathcal{H}$ as follows:

$$p(\mathbf{\Theta}|\mathcal{D},\mathcal{H}) = \frac{p(\mathcal{D}|\mathbf{\Theta})p(\mathbf{\Theta}|\mathcal{H})}{p(\mathcal{D}|\mathcal{H})} \tag{4.32}$$

As we stated earlier, the direct computation of posterior distribution $p(\mathbf{\Theta}|\mathcal{D},\mathcal{H})$ is computationally intractable [Mac02]. Therefore, we approximate $p(\mathbf{\Theta}|\mathcal{D},\mathcal{H})$ with a surrogate density $q(\mathbf{\Theta})$, found by maximising a variational lower bound to $log\big(p(\mathcal{D})\big)$ [WMR96]. In order to find an approximate posterior distribution over $\mathbf{\Theta}$ ($q(\mathbf{\Theta})$), we seek $q(\mathbf{\Theta})$ that will minimise the Kullback-Leibler divergence from the true posterior to $q(\mathbf{\Theta})$:

$$\mathbb{D}_{KL}\big(q(\mathbf{\Theta})||p(\mathbf{\Theta}|\mathcal{D},\mathcal{H})\big) \overset{\text{def}}{=} \int q(\mathbf{\Theta}) log \frac{q(\mathbf{\Theta})}{p(\mathbf{\Theta}|\mathcal{D},\mathcal{H})} \tag{4.33}$$

Now, the divergence can be written in terms of log likelihood and the variational free energy, $\mathcal{F}\big(q(\mathbf{\Theta})\big)$ ([Bis06]), as

$$log\, p(\mathcal{D}|\mathbf{\Theta},\mathcal{H}) = \mathbb{D}_{KL}\big(q(\mathbf{\Theta})||p(\mathbf{\Theta}|\mathcal{D},\mathcal{H})\big) + \mathcal{F}\big(q(\mathbf{\Theta})\big) \tag{4.34}$$

where

$$\mathcal{F}\big(q(\mathbf{\Theta})\big) \overset{\text{def}}{=} \int q(\mathbf{\Theta}) log \frac{p(\mathbf{\Theta},\mathcal{D}|\mathcal{H})}{q(\mathbf{\Theta})} d\mathbf{\Theta} \tag{4.35}$$

From Eq. (4.35), we can minimise Eq. (4.34) by maximising $\mathcal{F}\big(q(\mathbf{\Theta})\big)$ with respect to the choice of $q(\mathbf{\Theta})$. Additionally, $\mathcal{F}\big(q(\mathbf{\Theta})\big)$ serves as a lower bound to the log marginal likelihood. Now, our variational objective is to maximise $\mathcal{F}\big(q(\mathbf{\Theta})\big)$ with respect to $q(\mathbf{\Theta})$. However, maximising $\mathcal{F}(q\big(\mathbf{\Theta})\big)$ is minimizing $\displaystyle\int q_j\ log\ q_j\ d\mathbf{\Theta}_j$ where q_j is the j^{th} distribution in $q\big(\mathbf{\Theta}\big)$ [Bis06]. And, $\displaystyle\int q_j\ log\ q_j\ d\mathbf{\Theta}_j$ is minimum when:

$$\begin{aligned} log\big(q_j\big) &= \mathbb{E}_{i\neq j}\Big[log(p(\mathbf{\Theta},\mathcal{D}|\mathcal{H}))\Big]\ +\ const \\ q_j &= \frac{exp\Big(\mathbb{E}_{i\neq j}\big[log(p(\mathbf{\Theta},\mathcal{D}|\mathcal{H}))\big]\Big)}{\int exp\Big(\mathbb{E}_{i\neq j}\big[log(p(\mathbf{\Theta},\mathcal{D}|\mathcal{H}))\big]d\mathbf{\Theta}_j\Big)} \end{aligned}$$

From the above equation, we need to choose $p(\mathbf{\Theta})$ and define $log\big(p(\mathbf{\Theta},\mathcal{D})\big)$. The joint density, $\big(p(\mathbf{\Theta},\mathcal{D})\big)$, in the numerator of our objective function in Eq. 4.35 is written in the following way:

$$\begin{aligned} p(\mathcal{D},\mathbf{\Theta}|\mathcal{H}) = \prod_{m=1}^{M}\prod_{n=1}^{N}&\Bigg[p(r_{mn}|\hat{r}_{mn},\mathbf{J})p(\hat{r}_{mn}|\mathbf{z}_{mn})\prod_{k=1}^{K}\prod_{l=1}^{L}p(z_{mn}^{kl}|\alpha_k^m,\beta_l^n,\gamma_k^l,\delta_l^k) \\ &\cdot\prod_{k=1}^{K}p(x_k^m|\alpha_k^m,\Lambda_k)\prod_{l=1}^{L}p(y_l^n|\beta_l^n,\Lambda_l)\Bigg] \\ &\cdot\prod_{m=1}^{M}\prod_{n=1}^{N}\Bigg[\prod_{k=1}^{K}\mathcal{B}(\alpha_k^m\,;\,\Theta^{\alpha_k^m})\prod_{l=1}^{L}\mathcal{B}(\beta_l^n\,;\,\Theta^{\beta_l^n})\Bigg] \\ &\cdot\prod_{k=1}^{K}\prod_{l=1}^{L}\mathcal{B}(\gamma_k^l\,;\,\Theta_k^l)\mathcal{B}(\delta_l^k\,;\,\Theta_l^k) \\ &\cdot\prod_{k=1}^{K}p(\Lambda_k^0)p(\Lambda_k^1)\prod_{l=1}^{L}p(\Lambda_l^0)p(\Lambda_l^1) \end{aligned}$$

We assume that $q(\mathbf{\Theta})$ fully factorises over the the model parameters where:

$$\begin{aligned} q(\mathbf{\Theta}) = \prod_{m=1}^{M}\prod_{n=1}^{N}&\Bigg[q(r_{mn})\prod_{k=1}^{K}\prod_{l=1)}^{L}q(z_{mn}^{kl})\cdot\prod_{k=1}^{K}q(x_k^m)\prod_{l=1}^{L}q(y_l^n)\Bigg] \\ &\cdot\prod_{m=1}^{M}\prod_{n=1}^{N}\Bigg[\prod_{k=1}^{K}q(\alpha_k^m)\prod_{l=1}^{L}q(\beta_l^n)\Bigg] \\ &\cdot\prod_{k=1}^{K}\prod_{l=1}^{L}q(\gamma_k^l)q(\delta_l^k) \\ &\cdot\prod_{k=1}^{K}q(\Lambda_k^0)q(\Lambda_k^1)\prod_{l=1}^{L}q(\Lambda_l^0)q(\Lambda_l^1) \end{aligned}$$

In the following subsections, we approximate each factor in $q(\mathbf{\Theta})$ with simple exponential family distributions and provide a probability estimation for latent variable z_{mn}^{kl}. However, the subsequent derivation of update parameters for each factor probability distribution in $q(\mathbf{\Theta})$ by maximising the $log(q(\mathbf{\Theta}))$ with respect to the factor's distributional parameters are provided in Appendix A.

The approximate distribution of $q(x_k^m)$ and $q(y_l^n)$

In our present model, x_k^m and y_l^n are either ratings (in the CF case) or frequencies. Therefore, we assume that the x_k^m is a sample from a Poisson distribution with mean λ_k^0 if $\mathcal{O}_m$ belongs to the set of information objects of $\mathcal{A}$ which are described (with binary feature value "1" or elite to the feature) by the k^{th} feature in $\mathbf{E}$, and sample another Poisson distribution with mean λ_k^1 if $\mathcal{O}_m$ belongs to the set of objects that are not described by the k^{th} feature in $\mathbf{E}$. However, we don't know the sampled distribution. Therefore, we assume that the the observed x_k^m is a sample from a 2-Poisson mixture of two distributions corresponding to the information objects in $\mathcal{A}$. As previously described in Chapter 3, this is a 2-Poisson model assumption on the distribution of observed feature values in information objects. Similarly, we assume that the observed y_l^n is also a sample from a 2-Poission mixture corresponding to the information objects in $\mathcal{B}$ with means λ_l^0 and λ_l^1 respectively. We represent these means with sets $\Lambda_k = \{\lambda_k^0, \lambda_k^1\}$ and $\Lambda_l = \{\lambda_l^0, \lambda_l^1\}$.

Based on these assumption we write the approximate q distributions as follows:

$$\begin{aligned}
q(x_k^m) = p(x_k^m|\alpha_k^m, \Lambda_k) &= \alpha_k^m Pos(\lambda_k^0) + (1-\alpha_k^m)Pos(\lambda_k^1) \\
&= \alpha_k^m \Big(\frac{e^{-\lambda_k^0}(\lambda_k^0)^{x_k^m}}{x_k^m!}\Big) + (1-\alpha_k^m)\Big(\frac{e^{-\lambda_k^1}(\lambda_k^1)^{x_k^m}}{x_k^m!}\Big) \\
&= \left[\frac{e^{-\lambda_k^0}(\lambda_k^0)^{x_k^m}}{x_k^m!}\right]^{\omega_0} \left[\frac{e^{-\lambda_k^1}(\lambda_k^1)^{x_k^m}}{x_k^m!}\right]^{1-\omega_0} \\
q(y_l^n) = p(y_l^n|\beta_l^n, \Lambda_l) &= \beta_l^n Pos(\lambda_l^0) + (1-\beta_l^n)Pos(\lambda_l^1) \\
&= \beta_l^n \Big(\frac{e^{-\lambda_l^0}(\lambda_l^0)^{y_l^n}}{y_l^n!}\Big) + (1-\beta_l^n)\Big(\frac{e^{-\lambda_l^1}(\lambda_l^1)^{y_l^n}}{y_l^n!}\Big) \\
&= \left[\frac{e^{-\lambda_l^0}(\lambda_l^0)^{y_l^n}}{y_l^n!}\right]^{\omega_1} \left[\frac{e^{-\lambda_l^1}(\lambda_l^1)^{y_l^n}}{y_l^n!}\right]^{1-\omega_1},
\end{aligned}$$

where the given α_k^m and β_l^n indicate that the observed x_k^m and y_l^n values are generated from Poisson distributions of the k^{th}, l^{th} features corresponding to their frequency distributions in the collection of information objects they describe. Additionally, the given $(1-\alpha_k^m)$ and $(1-\beta_l^n)$ describe the opposite. Finally, we choose gamma distribution as a prior for each λ as gamma is a conjugate prior to Poisson distribution with two hyper parameters. Therefore, we set a gamma prior on each λ with hyper parameters (c, d). The c, d indices indicate the corresponding distribution to which they belong. For example (c_k^0, d_k^0) are the gamma distribution hyper parameters for the prior distribution on λ_k^0. See Appendix A for more details.

The approximate distribution of $q(\alpha_k^m), q(\beta_l^n), q(\gamma_k^l)$ and $q(\delta_l^k)$

Based on the probabilistic IMF model described in section 3.5.1 of Chapter 3, we know that α_k^m, β_l^n are binary random variables representing whether a k^{th} feature in $\mathbf{E}$ and an l^{th} feature in $\mathbf{F}$ describe the given information objects $\mathcal{O}_m$ and $\mathcal{O}_n$ respectively. Similarly, γ_k^l, δ_l^k are binary random variables representing the preference mappings between the k^{th} and l^{th} features respectively. We assume each of these binary random variables are samples from a beta distribution with their corresponding beta distribution hyper parameters represented with (θ^1, θ^2). Now, the parameters under the approximated q distributions can be expressed using their beta distributions as follows: The subscripts of each hyper

Approximated parameter distributions
$\alpha_k^m \sim \mathcal{B}(\theta^1_{\alpha_k^m}, \theta^2_{\alpha_k^m}) = {\alpha_k^m}^{\theta^1}(1-\alpha_k^m)^{\theta^2_{\alpha_k^m}}$
$\beta_l^n \sim \mathcal{B}(\theta^1_{\beta_l^n}, \theta^2_{\beta_l^n}) \ = {\beta_l^n}^{\theta^1_{\beta_l^n}}(1-\beta_l^n)^{\theta^2_{\beta_l^n}}$
$\gamma_k^l \sim \mathcal{B}(\theta^1_{\gamma_k^l}, \theta^2_{\gamma_k^l}) \ = {\gamma_k^l}^{\theta^1}(1-\gamma_k^l)^{\theta^2_{\gamma_k^l}}$
$\delta_l^k \sim \mathcal{B}(\theta^1_{\delta_l^k}, \theta^2_{\delta_l^k}) \ = {\delta_l^k}^{\theta^1_{\delta_l^k}}(1-\delta_l^k)^{\theta^2_{\delta_l^k}}$

parameter indicates their corresponding variable.

The approximate distribution of $q(z_{mn}^{kl})$ and $q(r_{mn})$

We know that the binary latent variable z_{mn}^{kl} indicates whether the bi-directional preferences between two information objects, $\mathcal{O}_m$ and $\mathcal{O}_n$ are satisfied with respect to the k^{th} and the l^{th} features respectively. We write the probability of the latent random variable z_{mn}^{kl} under the approximated q distribution as follows:

$$\begin{aligned} p(z_{mn}^{kl}|\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k) = & \left(\left[1-(1-\alpha_k^m)\beta_l^n\delta_l^k\right]\left[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\right] \right)^{z_{mn}^{kl}} \\ & \left(1 - \left(\left[1-(1-\alpha_k^m)\beta_j^n\delta_l^k\right]\left[1-(1-\beta_l^n)\alpha_k^m\gamma_k^l\right]\right)\right)^{\left(1-z_{mn}^{kl}\right)} \end{aligned} \tag{4.36}$$

In order to derive optimisation parameters, we apply a few algebraic transformations and apply data augmentation. The data augmentation is a method for constructing iterative optimisation procedure by introducing unobserved data or latent variables. We rewrite Eq. 4.36 by applying few transformations as follows:

$$\begin{aligned} p(z_{mn}^{kl}|\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k) = & \Big[\alpha_k^m(1-\beta_l^n)(1-\gamma_k^l) + (1-\alpha_k^m)(1-\delta_l^k)\beta_l^n + \alpha_k^m\beta_l^n + (1-\alpha_k^m)(1-\beta_l^n)\Big]^{z_{mn}^{kl}} \\ & .\Big[(1-\alpha_k^m)\beta_l^n\delta_l^k + (1-\beta_l^n)\alpha_k^m\gamma_k^l\Big]^{(1-z_{mn}^{kl})} \end{aligned} \tag{4.37}$$

Now, we convert the above sums into a product through data augmentation by introducing two binary latent variables ω_k^m and ω_l^n corresponding to α_k^m and β_l^n where $\omega_k^m, \omega_l^n \in \{0, 1\}$. We rewrite Eq. 4.37 by applying data augmentation as follows:

$$\begin{aligned} p(z_{mn}^{kl}|\alpha_k^m, \beta_l^n, \gamma_k^l, \delta_l^k) = & \left[\left[\left(1-\gamma_k^l\right)^{(1-\omega_l^n)}\right]^{\omega_k^m}\left[\left(1-\delta_l^k\right)^{\omega_l^n}\right]^{(1-\omega_k^m)}\right]^{z_{mn}^{kl}} \\ & .\left[\left(\beta_l^n\delta_l^k\right)^{1-\omega_k^m}\left((1-\beta_l^n)\gamma_k^l\right)^{\omega_k^m}\right]^{(1-z_{mn}^{kl})} \end{aligned}$$

Finally, we assume that the ω_k^m and ω_l^n follows Bernoulli distributions with probability α_k^m and β_l^n respectively. And their probability is defined as follows:

$$p(\omega_k^m|\alpha_k^m) = (\alpha_k)^{\omega_k^m}(1-\alpha_k^m)^{1-\omega_k^m} \quad \text{and} \quad p(\omega_l^n|\beta_l^n) = (\beta_l^n)^{\omega_l^n}(1-\beta_l^n)^{1-\omega_l^n}$$

Once we know the binary vector $\mathbf{z}_{mn}$, it is easy to verify the relevance between the $\mathcal{O}_m$ and $\mathcal{O}_n$ pair as every component in $\mathbf{z}_{mn}$, i.e. z^{kl}_{mn}, must be "1". We take this fact and define the relevance under the q distribution given $\mathbf{z}_{mn}$ as follows:

$$\begin{aligned} q(r_{mn}) = p(r_{mn}|\mathbf{z}_{mn}) &\sim Bern\Big(\prod_{k,l} z^{k,l}_{mn}, 1 - \prod_{k,l} z^{k,l}_{mn}\Big) \\ &= \left(\prod_{k,l} z^{k,l}_{mn}\right)^{r_{mn}} \left(1 - \prod_{kl} z^{k,l}_{mn}\right)^{1-r_{mn}} \end{aligned}$$

where

$$\prod_{k,l} z^{k,l}_{mn} \sim \quad \mathcal{B}(\Theta^n_m) = \left(\prod_{k,l} z^{k,l}_{mn}\right)^{\theta^n_m(1)} \left(1 - \prod_{k,l} z^{k,l}_{mn}\right)^{\theta^n_m(2)},$$

and $\big(\theta^n_m(1), \theta^n_m(2)\big)$ are hyper parameters of a beta distribution.

The fully factorised approximated distribution

We define the final factorised approximated distribution, $q(\mathbf{\Theta})$, by adding the ω variables as follows:

$$\begin{aligned} q(\mathbf{\Theta}) = \prod_{m=1}^{M}\prod_{n=1}^{N} &\left[q(r_{mn}) \prod_{k=1}^{K}\prod_{l=1}^{L} q(z^{kl}_{mn}) \cdot \prod_{k=1}^{K} q(x^m_k) q(\omega^m_k) \prod_{l=1}^{L} q(y^n_l) q(\omega^n_l)\right] \\ \cdot \prod_{m=1}^{M}\prod_{n=1}^{N} &\left[\prod_{k=1}^{K} q(\alpha^m_k) \prod_{l=1}^{L} q(\beta^n_l)\right] \cdot \prod_{k=1}^{K}\prod_{l=1}^{L} q(\gamma^l_k) q(\delta^k_l) \cdot \prod_{k=1}^{K} q(\Lambda^0_k) q(\Lambda^1_k) \prod_{l=1}^{L} q(\Lambda^0_l) q(\Lambda^1_l) \end{aligned}$$

As stated earlier, we can estimate each factor's distributional parameters by minimising the objective function $\mathcal{F}(q\big(\mathbf{\Theta}\big)$ with respect to it's corresponding factor distribution where $\mathcal{F}(q\big(\mathbf{\Theta}\big)) = log(q(\mathbf{\Theta}))$. We minimise it by computing the functional derivatives $\frac{\partial \mathcal{F}}{\partial q}$ with respect to each distribution in $q(\mathbf{\Theta})$ and setting it equal to zero. By setting the derivatives to zero we get the update steps for each factor distribution in $q(\mathbf{\Theta})$ in terms of their sufficient statistics. The derivations for estimating these parameters are presented in Appendix A. In general we can categorise these parameters of the graphical model from the section 3.5.1 into two classes: (a) model parameters described in Table 4.2 and (b) data augmented binary latent variables in Table 4.3.

Parameter estimation in the Graphical Model

In order to estimate the optimal parameter values, we update each parameter iteratively by fixing the others. The basic algorithm for learning the optimal parameters in the IMF model is outlined in algorithm 3 below. The summary of the derived update equations for the parameters of the exponential family factor distributions in q are given in Table 4.4.

Ranking Score

Once the parameters are learned, we can compute the relevance ranking score by setting $\forall k, l\ z^{kl}_{mn} = 1$ ($r_{mn} = 1 \Longrightarrow \forall k, l\ z^{kl}_{mn} = 1$). By substituting z^{kl}_{mn} values into our relevance hypothesis, we get:

$$\begin{aligned} p(r_{mn} = 1|\mathcal{O}_m, \mathcal{O}_n) \propto_r \prod_{k=1}^{K}\prod_{l=1}^{L} \Big[&\alpha^m_k(1-\beta^n_l)(1-\gamma^l_k) + (1-\alpha^m_k)(1-\delta^k_l)\beta^n_l + \\ &\alpha^m_k\beta^n_l + (1-\alpha^m_k)(1-\beta^n_l)\Big] \end{aligned} \tag{4.38}$$

α_k^m	Prior	$\theta^1_{\alpha_k^m}$ and $\theta^2_{\alpha_k^m}$
	Posterior	$\theta^1_{\alpha_k^m}(new) = \langle \omega_k^m \rangle_q + \theta^1_{\alpha_k^m}$ and $\theta^2_{\alpha_k^m}(new) = (1 - \langle \omega_k^m \rangle_q) + \theta^2_{\alpha_k^m}$
	Expected value	$\langle log\ \alpha_k^m \rangle_q = \Psi\big(\theta^1_{\alpha_k^m}(new)\big) - \Psi\big(\theta^1_{\alpha_k^m}(new) + \theta^2_{\alpha_k^m}(new)\big)$ and $\langle log(1-\alpha_k^m) \rangle_q = \Psi\big(\theta^2_{\alpha_k^m}(new)\big) - \Psi\big(\theta^1_{\alpha_k^m}(new) + \theta^2_{\alpha_k^m}(new)\big)$
β_l^n	Prior	$\theta^1_{\beta_l^n}$ and $\theta^2_{\beta_l^n}$
	Posterior	$\theta^1_{\beta_l^n}(new) = \langle \omega_l^n \rangle_q + \theta^1_{\beta_l^n} + \sum_m \sum_k (1 - \left\langle z_{mn}^{kl} \right\rangle_q)(1 - \langle \omega_k^m \rangle_q)$ and $\theta^2_{\beta_l^n}(new) = \theta^2_{\beta_l^n} + (1 - \langle \omega_l^n \rangle_q) + \sum_m \sum_k (1 - \left\langle z_{mn}^{kl} \right\rangle_q) \langle \omega_k^m \rangle_q$
	Expected value	$\langle log\beta_l^n \rangle_q = \Psi\big(\theta^1_{\beta_l^n}(new)\big) - \Psi\big(\theta^1_{\beta_l^n}(new) + \theta^2_{\beta_l^n}(new)\big)$ and $\langle log(1-\beta_l^n) \rangle_q = \Psi\big(\theta^2_{\beta_l^n}(new)\big) - \Psi\big(\theta^1_{\beta_l^n}(new) + \theta^2_{\beta_l^n}(new)\big)$
γ_k^l	Prior	$\theta^1_{\gamma_k^l}$ and $\theta^2_{\gamma_k^l}$
	Posterior	$\theta^1_{\gamma_k^l}(new) = \sum_m \sum_n (1 - \left\langle z_{mn}^{kl} \right\rangle_q) \langle \omega_k^m \rangle_q + \theta^1_{\gamma_k^l}$ and $\theta^2_{\gamma_k^l}(new) = \sum_m \sum_n \left\langle z_{mn}^{kl} \right\rangle_q \left[\langle \omega_k^m \rangle_q (1 - \langle \omega_l^n \rangle_q) \right] + \theta^2_{\gamma_k^l}$
	Expected value	$\left\langle log(\gamma_k^l) \right\rangle_q = \Psi(\theta^1_{\gamma_k^l}(new)) - \Psi\big(\theta^1_{\gamma_k^l}(new) + \theta^2_{\gamma_k^l}(new)\big)$ and $\left\langle log(1-\gamma_k^l) \right\rangle_q = \Psi(\theta^2_{\gamma_k^l}(new)) - \Psi\big(\theta^1_{\gamma_k^l}(new) + \theta^2_{\gamma_k^l}(new)\big)$
δ_l^k	Prior	$\theta^1_{\delta_l^k}$ and $\theta^2_{\delta_l^k}$
	Posterior	$\theta^1_{\delta_l^k}(new) = \sum_m \sum_n (1 - \left\langle z_{mn}^{kl} \right\rangle_q)(1 - \langle \omega_k^m \rangle_q) + \theta^1_{\delta_l^k}$ and $\theta^2_{\delta_l^k}(new) = \sum_m \sum_n \left\langle z_{mn}^{kl} \right\rangle_q \left[(1 - \langle \omega_k^m \rangle_q) \langle \omega_l^n \rangle_q \right] + \theta^2_{\delta_l^k}$
	Expected value	$\left\langle log(\delta_l^k) \right\rangle_q = \Psi(\theta^1_{\delta_l^k}(new)) - \Psi\big(\theta^1_{\gamma_k^l}(new) + \theta^1_{\delta_l^k}(new)\big)$ and $\left\langle log(1-\delta_l^k) \right\rangle_q = \Psi(\theta^2_{\delta_l^k}(new)) - \Psi\big(\theta^1_{\delta_l^k}(new) + \theta^2_{\delta_l^k})(new)\big)$
λ_0^k and λ_1^k	Prior	a_0^k, b_0^k, a_1^k and b_1^k
	Posterior	$a_0^k(new) = a_0^k - 1 + \sum_m \langle \omega_k^m \rangle_q x_k^m$, $b_0^k(new) = b_0^k + \sum_m \langle \omega_k^m \rangle_q$, $a_1^k(new) = \left[a_1^k - 1 + \sum_m x_k^m (1 - \langle \omega_k^m \rangle_q)\right]$ and $b_1^k(new) = b_1^k + \sum_m (1 - \langle \omega_k^m \rangle_q)$
	Expected value	$\left\langle log\ \lambda_0^k \right\rangle_q = \Psi(a_0^k(new)) - log(b_0^k(new))$ and $\left\langle log\lambda_1^k \right\rangle_q = \Psi(a_1^k(new)) - log(b_1^k(new))$
λ_0^l and λ_1^l	Prior	a_0^l, b_0^l, a_1^l and b_1^l
	Posterior	$a_0^l(new) = \left[a_0^l - 1 + \sum_n \langle \omega_l^n \rangle_q y_l^n\right]$, $b_0^l(new) = b_0^l + \sum_n \langle \omega_l^n \rangle_q$ and $a_1^l(new) = \left[a_1^l - 1 + \sum_n y_l^n (1 - \langle \omega_l^n \rangle_q)\right]$, $b_1^l(new) = \mathbf{b}_1^l + \sum_n (1 - \langle \omega_l^n \rangle_q)$
	Expected value	$\left\langle log\lambda_0^l \right\rangle_q = \Psi(a_0^l(new)) - log(b_0^l(new))$ and $\left\langle log\lambda_1^l \right\rangle_q = \Psi(a_1^l(new)) - log(a_1^l(new))$

Table 4.4: IMF approximated parameter update equations for each factor distribution in q. Ψ is a digamma function and defined as $\Psi(c) = \dfrac{d}{dc}\ ln(\Gamma(c))$ and the expected value is the expected estimated log probability of the variable under approximated distribution q.

Algorithm 3 Learning Algorithm

```
Each update step is computed in parallel
Initialise all the priors
for $i = 1, \cdots, \#iterations$ do
    $\forall k, m$ update hyper parameters of $\alpha_k^m$
    $\forall l, n$ update hyper parameters of $\beta_l^n$
    $\forall k, m$ update $\omega_k^m$
    $\forall l, n$ update $\omega_l^n$
    $\forall k, l$ update forward mapping parameters $\gamma_k^l$
    $\forall k, l$ update backward mapping parameters $\delta_l^k$
    $\forall k, l$ update hyper parameters of all $\lambda^k$'s and $\lambda^l$'s
    $\forall m, n$ update $\mathbf{z}_{mn}$
end for
```

	item $\mathbf{v}_1$ ($\mathcal{O}_n^1$)	item $\mathbf{v}_2$ ($\mathcal{O}_n^2$)
user $\mathbf{u}_1$ ($\mathcal{O}_m^1$)	5	0
user $\mathbf{u}_2$ ($\mathcal{O}_m^2$)	0	5

Table 4.5: An example user-item rating matrix.

All the parameters of the probabilistic relevance function in Eq. (4.38) that are associated with information objects $\mathcal{O}_m$ (α_k^m's) and $\mathcal{O}_n$ (β_l^n's), and the mappings γ_k^l's and δ_l^k's are learned through the approximate inference method presented in the previous section. In the following section, we illustrate the optimisation procedure presented in this section through a simple example from Chapter 3.

Example

Let us consider the following example of a recommendation scenario. Assume that we have two users with orthogonal tastes, i.e. if one user likes an item then the second user does not, and vice-versa. We can represent this scenario with a simple 2×2 user-item rating matrix where both users have assigned complement ratings as shown in Table. 4.5. The data clearly shows (based on the numerical rating) that $\mathbf{u}_1$ *liked* the item $\mathbf{v}_1$ more compared to the item $\mathbf{v}_2$, and $\mathbf{u}_2$ *liked* the item $\mathbf{v}_2$ more than the item $\mathbf{v}_1$.

If our IMF model is applied to any such users' dataset, it has to learn their preferences in similar fashion. In order to apply IMF to this problem, we first need to define the feature space for both user and item representation. Let us define two user description features for representing the users, each corresponding to one item in our data set. Now, according to the model, the user rating on an item depends on the corresponding item's binary description feature value for the user (i.e. if the eliteness (or binary feature value) is "1" we observe a higher rating). Similarly, we define two features, each corresponding to one user, which describe items. Now, the item's received rating from a user is dependent on the corresponding feature description value for the feature represented by the user. For example, from the user-item rating matrix, $\mathbf{u}_1$ is described by the feature corresponding to the item $\mathbf{v}_1$ and not described

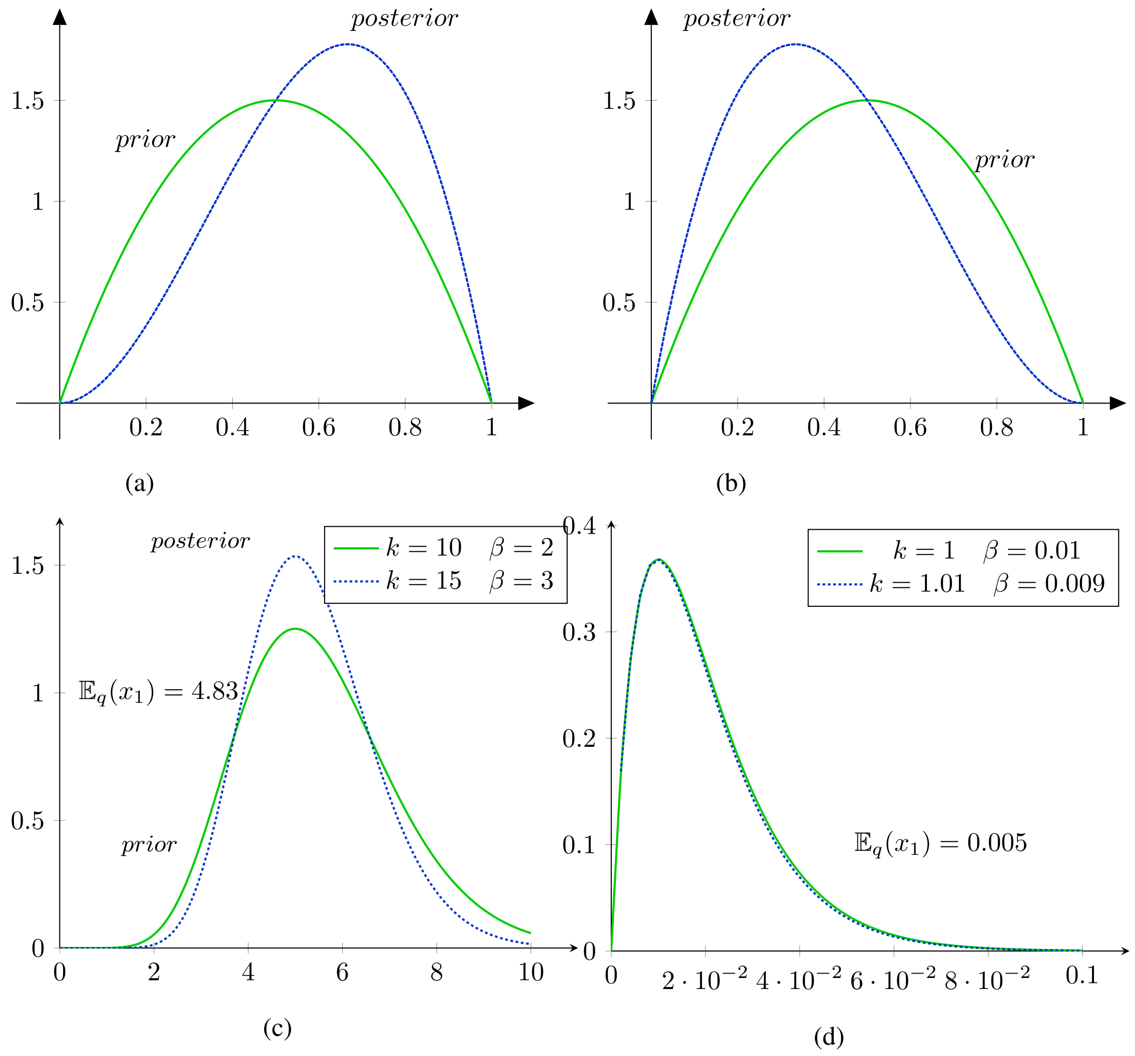


Figure 4.4: (a) The prior and the posterior of distribution of the probability density function of α_1^1. (b) The prior and the posterior of the probability density function of α_1^2. (c) The prior and posterior probability density function representing average of the mean rating of the feature corresponding to $\mathbf{v}_1$ in the objects it describes (*elite* set). (d) The prior and posterior probability density function representing the average mean rating of feature corresponding to $\mathbf{v}_1$ in a objects it does not describes (*elite* set).

by feature corresponding to $\mathbf{v}_2$ ($r_{11} > r_{12}$). Therefore, when we apply the learning procedure, the model has to learn this observation.

After applying the learning procedure to this data set, the probability density functions of α_1^1 and α_1^2 distributions are shown in (a) and (b) of Figure 4.4. For the user $\mathbf{u}_1$, the distributions α_1^1 and α_1^2 represents the uncertainty in features corresponding to the items $\mathbf{v}_1$ and $\mathbf{v}_2$. From Figure 4.4, we can observe that the probability of describing the user $\mathbf{u}_1$ with the feature corresponding to $\mathbf{v}_1$ is greater than feature corresponding to item $\mathbf{v}_2$. In other words, from the distribution graphs in (c) and (d) of Figure 4.4, we can see that $\mathbb{E}_q(x_1) << \mathbb{E}_q(x_0)$ describing that of the expected average value of the feature for an item corresponding to the feature that describes it is greater than that of the feature that does not describe it. Similarly, we observe the opposite in the case of $\mathbf{u}_2$. Therefore, the model learns the clear distinction in features representing each object, and the model-learnt distributions represent our observations. Similarly, Figure 4.5 shows that the ω_1^1 and ω_1^2 are learned as desired where ω_1^1 and ω_1^2 approach desired values of 1 and 0 respectively. That is, after learning, $\omega_1^1 \approx 1$ and $\omega_1^2 \approx 0$. Finally, Figure 4.7 shows that the model converges to a stable point after 20 iterations. The error in this particular case is computed as follows: $error = \frac{1}{4}\sum_{i=1}^{2}\sum_{j=1}^{2}|\hat{r}_{ij} - r_{ij}|$ where $\hat{r}_{ij} = 1$ if $rating \geq 2$, "0" otherwise and r_{ij} is estimated probability of relevance. In conclusion, from the simple example, we can see that the model learns the desired parameter values for representing the features and mappings after a few iterations. However, in the final section of this chapter, we will discuss some drawbacks in using this complete model for optimisation.

As of now, we have seen the inference in the IMF model (bi-directional unified model). In the following section, we present optimisation methods for estimating the 2-Poisson mixture parameters. Note that the 2-Poisson mixture model parameter estimation is the same for computing the probability in both the CF and ad-hoc retrieval tasks in sections 4.2.4 and 4.4.4. Therefore, we present the parameter estimation only once and use in both cases.

4.5.2 Estimating the 2-Poisson mixture parameters

In this section, we present two parameter estimation methods for the 2-Poisson mixture. In section 4.4.4, in order to estimate $p(tf_i|f_i = 1)$ and $p(tf_i|f_i = 0)$, we assumed that the frequency of a term, tf, or the rating of a user/item follows a Poisson distribution in a collection of documents/users/items where the corresponding term-description (or user/item description) feature is *elite* (the binary feature value is "1"), and a different Poisson distribution in a collections where the feature is not elite. This assumption was made based on the successful 2-Poisson model hypothesis [Har75a, RvRP80].

Let x_i be the corresponding observed feature value of the feature s_i where $s_i \in \Phi$ and $\alpha_i \in \{0, 1\}$ be the binary feature value of s_i. Now, from the above assumption, we can write:

$$p(x_i|\alpha_i = 1) := e^{-\mu_i(1)}\mu_i(1)^{x_i}$$
$$p(x_i|\alpha_i = 0) := e^{-\mu_i(0)}\mu_i(0)^{x_i}$$

where $\mu_i(1), \mu_i(0)$ are the means of the corresponding Poisson distributions and x_i is the term frequency or rating depending on the application. According to the 2-Poisson mixture model, $p(x_i)$ is a mixture of

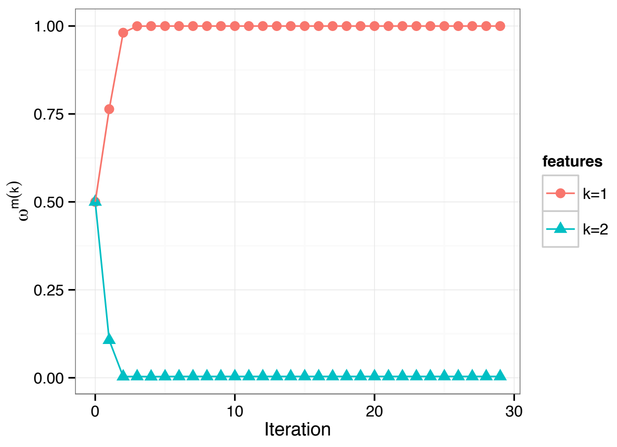


Figure 4.5: The convergence of ω_1^1 and ω_1^2.

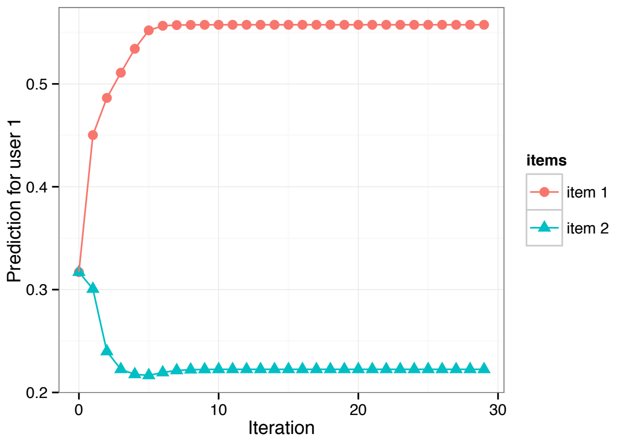


Figure 4.6: The user u_1 relevance to the items $\mathbf{v}_1$ and $\mathbf{v}_2$.

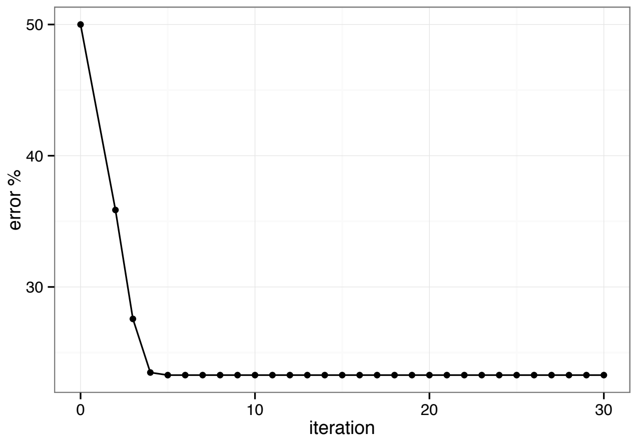


Figure 4.7: The model training error for each iteration.

two Poisson distributions. Let p_i be the mixing probability corresponding to i then $p(\alpha_i = 1) := p_i$ and $p(\alpha_i = 0) := 1 - p_i$. The 2-Poisson mixture can be written as

$$\begin{aligned} p(x_i|\theta_i) = & \sum_{\alpha_i \in \{0,1\}} p(x_i|\alpha_i)p(\alpha_i) \\ = & p_i \frac{e^{-\mu_i(1)}\mu_i(1)^{x_i}}{x_i!} + (1-p_i)\frac{e^{-\mu_i(0)}\mu_i(0)^{x_i}}{x_i!} \end{aligned} \tag{4.39}$$

where θ_i is the unknown parameters of the mixture and $\theta_i = (\mu_i(1), \mu_i(0), p_i)$. We estimate the optimal parameter values of the mixture by using maximum likelihood estimation (using the Expectation Maximization (EM) algorithm [DLR77]) as well as Gibbs sampling for finite mixtures via MCMC [DR94]. In the following we describe the estimation using both approaches.

EM algorithm

Let $\mathbf{x}_i$ be the vector where $\mathbf{x}_i = \langle x_i^1, x_i^2, \cdots x_i^N \rangle$ where $x_i^n \;\; (n \in [1, N])$ is the observed x_i for the n^{th} document/user/item and N is the total number of documents (items or users). Now, we can write the probability of observing x_i by assuming the independence as follows:

$$p(\mathbf{x}_i|\theta_i) = \prod_{n=1}^{N} p(x_i^n|\theta_i) \tag{4.40}$$

Now, the likelihood function as a function of θ_i can be written as,

$$\begin{aligned} \mathcal{L}(\theta_i) &= p(\mathbf{x}_i|\theta_i) \\ &= \prod_{n=1}^{N} p(x_i^n|\theta_i) \\ &= \prod_{n=1}^{N} \Big(p_i \frac{e^{-\mu_i(1)}\mu_i(1)^{x_i^n}}{x_i^n!} + (1-p_i)\frac{e^{-\mu_i(0)}\mu_i(0)^{x_i^n}}{x_i^n!} \Big) \end{aligned}$$

We obtain the Maximum Likelihood estimate of θ_i by maximising the log-likelihood $log\big(\mathcal{L}(\theta_i)\big)$ where

$$log\big(\mathcal{L}(\theta_i)\big) = \sum_{n=1}^{N} \log \left\{ p_i \frac{e^{-\mu_i(1)}\mu_i(1)^{x_i^n}}{x_i^n!} + (1-p_i)\frac{e^{-\mu_i(0)}\mu_i(0)^{x_i^n}}{x_i^n!} \right\}$$

Specially, we compute the partial derivatives of $log\big(\mathcal{L}(\theta_i)\big)$ with respect to $\mu_i(1)$, $\mu_i(0)$ and p_i, and set them to zero. By setting:

$$\frac{\partial \mathcal{L}}{\partial \mu_i(1)} = 0, \frac{\partial \mathcal{L}}{\partial \mu_i(0)} = 0 \tag{4.41}$$

we get,

$$\mu_i(1) = \frac{\sum_{n=1}^{N} p(\alpha_i = 1|x_i^n)\, x_i^n}{\sum_{n=1}^{N} p(\alpha_i = 1|x_i^n)} \tag{4.42}$$

and

$$\mu_i(0) = \frac{\sum_{n=1}^{N} p(\alpha_i = 0|x_i^n)\, x_i^n}{\sum_{n=1}^{N} P(\alpha_i = 0|x_i^n)}. \tag{4.43}$$

By setting

$$\frac{\partial \mathcal{L}}{\partial p_i} = 0 \tag{4.44}$$

we get,

$$p_i = \frac{1}{N} \sum_{n=1}^{N} p(\alpha_i = 1|x_i^n) \tag{4.45}$$

where $p(\alpha_i = 1|x_i^n)$ is computed as follows:

$$p(\alpha_i = 1|x_i^n) = \frac{p(x_i^n|\alpha_i = 1)p(\alpha_i = 1)}{\sum_{\alpha_i \in \{0,1\}} p(x_i^n|\alpha_i)p(\alpha_i)} \tag{4.46}$$

In order to implement the EM algorithm, we start with an initial an estimate of $p_i^{(0)}$, $\mu_i(1)^{(0)}$, $\mu_i(0)^{(0)}$ and then the better estimates can be found by iterating through the following E-step and M-step [DLR77].

1. E-step:

$$p^{(k+1)}(\alpha_i = 1|x_i^n) = \frac{p^{(k)}(x_i^n|\alpha_i = 1)p_i^{(k)}}{\sum_{\alpha_i \in \{0,1\}} p^{(k)}(\alpha_i|\alpha_i)p(\alpha_i)}. \tag{4.47}$$

2. M-step:

$$\begin{aligned}
\mu_i(0)^{(k+1)} &= \frac{\sum_{n=1}^{N} p^{(k)}(\alpha_i = 0|x_i^n)x_i^n}{\sum_{n=1}^{N} p^{(k)}(\alpha_i = 0|x_i^n)}. \\
\mu_i(1)^{(k+1)} &= \frac{\sum_{n=1}^{N} p^{(k)}(\alpha_i = 1|x_i^n)x_i^n}{\sum_{n=1}^{N} p^{(k)}(\alpha_i = 1|x_i^n)}. \\
p_i^{(k+1)} &= \frac{1}{N} \sum_{n=1}^{N} p^{(k)}(\alpha_i = 1|x_i^n).
\end{aligned}$$

where k is the step number. The stopping criteria for the EM is: if the difference between the perplexities of two consecutive models of the probability distribution $p(\alpha_i = 1|tf_i)$ is minute. We will describe the convergence of learning in ad-hoc retrieval experiments in Chapter 5.

Gibbs sampling

To estimate the 2-Poisson mixture parameters through Gibbs sampling via MCMC implementation, we formulate our sampling based on Bayesian sampling formulation for the finite mixtures proposed in [DR94].

Let us assume that $\mathbf{x}_i$ is the set of observed frequencies/ratings associated with the i^{th} feature in a specific type of object collection where $\mathbf{x}_i = \{x_i^1, \cdots, x_i^N\}$, and N is the total number of objects in the collection. An example object collection would be a set of documents or movies. Let θ_i be the set of parameters in the finite 2-Poisson mixture model where $\theta_i = (\mu_i, p_i)$, $\mu_i = \{\mu_i(0), \mu_i(1)\}$ is a set of Poisson means corresponding to the mixture distributions and $\pi(\theta_i)$ is the mixture component. Now, the posterior over the parameters in θ_i is written as follows:

$$p(\theta_i|x_i, \alpha_i) \propto p(x_i, \alpha_i|\theta)\pi(\theta_i)$$

where α_i describes the mixture component of the distribution (*elite* or *non-elite*). Before we sample parameters, we choose gamma distributions as a conjugate prior for each Poisson mean μ in θ_i and a beta conjugate prior for mixture probability component p_i. Given the prior distribution, π in parameter θ_i, Bayesian estimation is to simulate $\theta_i \approx p(\theta_i|x_i, \alpha_i)$. For the MCMC implementation, we need to sample from a conditional posterior distribution for the parameters $\mu_i = \{\mu_i(0), \mu_i(1)\}$, p_i and α_i (missing eliteness value). In order to estimate the optimal parameter values, we first initialize $\theta_i,\ \alpha$ and β, and the priors g and h and then iterate over the following sample steps until the model parameters (or Markov chain) converges to a stable point.

The sampling steps are:

1. $\alpha_i^n \approx$ Bernouli$(p_i(\alpha_i = 1|x_i^n))\ \forall n = 1$ to N where,

$$p_i(\alpha_i = 1|x_i^n) = p_i \,.\, \frac{p(x_i^n|\mu_i(1))}{p(x_i^n)}$$

   where $p(x_i^n) = p_i p(x_i^n|\mu_i(1)) + (1-p_i)p(x_i^n|\mu_i(0))$.

2.

$$p_i \approx \mathcal{B}\Big(a + \sum_{n=1}^{N} p(\alpha_i^n = 1|x_i^n), b + \sum_{n=1}^{N} p(\alpha_i^n = 0|x_i^n)\Big)$$
$$\mu_i(1) \approx gamma\Big(g + \sum_{n=1}^{N} \alpha_i^n x_i^n, h + \sum_{n=1}^{N} \alpha_i^n\Big)$$
$$\mu_i(0) \approx gamma\Big(g + \sum_{n=1}^{N} (1-\alpha_i^n) x_i^n, h + \sum_{n=1}^{N} (1-\alpha_i^n)\Big)$$

Now we can obtain optimal parameters for the 2-Poisson mixture model through either of the optimisation procedures presented in this section.

Ranking functions

We can substitute the optimal estimated parameter values for the 2-Poisson mixture model obtained in the above section into Eq. (4.16) and Eq. (4.23) to get the relevance ranking functions for collaborative filtering and group recommendation tasks. By substituting the optimal parameter values and applying log, we obtain the following final ranking functions for collaborative filtering and group recommendation,

Collaborative Filtering:

$$\begin{aligned} p(r=1|\mathbf{u}_m, \mathbf{v}_n) &\propto_r \sum_{\text{(k, l)}:r_{kl} \in \mathcal{D}^r} log\Big\{ \frac{p(rt_{mk}|\alpha_k = 1)}{\sum_{\alpha_k \in \{0,1\}} p(rt_{mk}|\alpha_k)p(\alpha_k)} \cdot \frac{p(rt_{nl}|\beta_l = 1)p(\beta_l = 1)}{\sum_{\beta_l \in \{0,1\}} p(rt_{ln}|\beta_l)} \Big\} \\ &\propto_r \sum_{\text{(k, l)}:r_{kl} \in \mathcal{D}^r} log\Big\{ \frac{e^{-\mu_k(1)}\mu_k(1)^{r_{kl}}}{p_i e^{-\mu_k(1)}\mu_k(1)^{r_{kl}} + (1-p_k)e^{-\mu_k(0)}\mu_k(0)^{r_{kl}}} \Big\} \\ &+ log\Big\{ \frac{e^{-\mu_l(1)}\mu_l(1)^{r_{kl}}}{p_l e^{-\mu_l(1)}\mu_l(1)^{r_{kl}} + (1-p_l)e^{-\mu_l(0)}\mu_l(0)^{r_{kl}}} \Big\} \end{aligned} \tag{4.48}$$

Group Recommendation:

$$p(r_g = 1|\mathcal{G}, \mathbf{v}_n) \propto_{r_g} \min\big\{p(r_1 = 1|\mathbf{u}_1, \mathbf{v}_n), \cdots, p(r_h = 1|\mathbf{u}_h, \mathbf{v}_n)\big\} \tag{4.49}$$

where $\forall\ h\ p(r_h = 1|\mathbf{u}_h, \mathbf{v}_n)$ is computed using the collaborative filtering ranking function.

The intuition behind the ranking function for the collaborative filtering in Eq (4.48) is given in section 4.2.3. And the ranking function corresponding to ad-hoc text retrieval is provided in Eq. 4.31

(page 78). In Chapter 5, we use the above ranking functions to demonstrate the applicability of the IMF in three different scenarios where three different types of information are available to the retrieval system.

4.6 Conclusion

Our primary objective for this chapter was to show that the same IMF and model can be applied in three different popular retrieval scenarios and to present an inference procedure for obtaining the optimal model parameters in the IMF graphical model. In order to apply IMF to different retrieval scenarios, we have shown that all we need is to: (a) identify two distinct feature spaces for probabilistically representing two sets of information objects, (b) define mappings between the features. However, defining mappings are rather straightforward once we define features in (a), and (c) identify interactive information between the objects such as a set of relevant pairs. We have shown that if we are able to execute these three steps, we can apply the IMF to any information matching task/application. However, there are cases where there is no interactive information between the objects, e.g., ad-hoc retrieval with no relevance judgements. In these cases, the application of the IMF is not straightforward as it relies on interaction information between the objects in estimating the preference mappings. And these mappings are crucial components of estimating the relevance. However, in these cases, we have shown how the IMF can be adopted by making some valid assumptions on preference mappings between the features describing the information objects to circumvent the problem of the lack of interaction information. In theory, as shown in this chapter, we can apply the IMF to number information matching tasks. Even though the IMF is flexible it poses enormous computational challenge as it can potentially use all the information objects in one class as features for representing the information objects in another class. For example, in the CF problem, a user feature space can include all the possible items. This will also lead to creating large mapping matrices of size $\mathbf{E} \times \mathbf{F}$ where $\mathbf{E}$ and $\mathbf{F}$ are two feature sets representing two distinct types of information objects and often $|\mathbf{E}|$ and $|\mathbf{F}|$ are in the millions in real-world web systems. Therefore, any application of the full model needs a careful consideration of the computational implications of the model, and often involves making approximations to the model. It remains a challenge to apply the IMF without excessive computational overheads.

In the next chapter, we run some experiments on test collections in three different retrieval scenarios to show the comparative performance of simple relevance ranking functions derived in this chapter. The primary objective of our experiments is to show that the IMF can be applied in practical settings and that it opens up further research for making performance-improving models based the new theory and the IMF presented in this thesis.

Chapter 5

Experiments

The primary objective for the thesis was to develop a theory and probabilistic model for solving the *problem of information matching* defined in Section 3.1.1. A necessary requirement of a solution for the problem is that, in computing the relevance between the information object pair $\mathcal{O}_m$ and $\mathcal{O}_n$, a proposed theory and model should be able to utilise all the available data on individual information objects ($\mathcal{O}_m$ and $\mathcal{O}_n$), the past interaction information on these objects (e.g., relevance) and any other information on the interaction of the other information objects in the collection ($\{\mathcal{A} \times \mathcal{B}\}$). In order to achieve this objective, in Chapter 3, we have proposed a new theory called the *theory of information matching* and developed an information matching framework (and a probabilistic model) based on a simple but powerful idea of *bi-directional preference mapping*. We referred to the framework as an *information matching framework (IMF)*.

In Chapter 4, we have demonstrated how the IMF can be applied to three different scenarios of available information. The first, (a), is where the available information is the interaction between information objects $\{\mathcal{A} \times \mathcal{B}\}$ and observed data on distinct feature spaces. There is no common feature space between the information objects of $\mathcal{A}$ and $\mathcal{B}$. This is a traditional collaborative filtering (recommendation) problem. The second, (b), is where an additional contextual information on the objects of $\mathcal{A}$ or $\mathcal{B}$ along with the information from (a). The second scenario corresponds to group recommendation. The third, (c), where there is an observed data on individual information objects in a common feature space (e.g. vocabulary) but there is no information available on interaction between the information objects. This corresponds to a simple ad-hoc retrieval task with no relevance information. We have also demonstrated, both theoretically and with some examples, on how the IMF can utilise information available from more than one scenario in computing the relevance.

It is also important to note that the core idea of *bi-directional preference mapping* is heavily reliant on the interaction information between the objects, as it learns the preference mappings between two distinct feature sets representing two distinct sets of objects using the interaction information. If there is interaction information, the IMF can be applied naturally. However, if there is no interactive information, it is hard to use the IMF without explicitly defining the mappings between the features. This is purely because the IMF does not assume a common feature space between the information objects, and there is no base to compare the information objects for relevance matching if there are no preference mappings

Scenario	Available information on objects	Application
1	Interaction (e.g., relevance) and descriptive features	Collaborative Filtering and Group Recommendation
2	Descriptive features with no interaction	Ad-hoc text retrieval (without relevance)
3	Interaction and descriptive features	A preference based movie search system

Table 5.1: A summary of applications chosen for experimentation corresponding to each scenario of available information.

between the features. The scenario (c) describes the lack of interaction information and showing how the IMF can be applied in this scenario ruled out an important outlier of not being able use the IMF.

The main purpose of this chapter is to pick an application from each of the above scenarios, run experiments on standard collections and demonstrate the applicability of the IMF through these experiments. The objective of our experiments is to show that the information matching theory and the IMF can be applied in a practical setting. However, it is important to note that the model cannot be seen in the light of performance in various evaluation metrics as it is fundamentally theoretical in nature and needs further research in making it efficient in terms of performance.

In order to apply the IMF to different scenarios, we choose three different applications based on the kind of information available, and summarised in Table 5.1. In the third application, we show how a preference based search engine can be built using our theory and framework by demonstrating a beta-release web preference search system based on our work and available online. It is important to note that our objective is to demonstrate the applicability of IMF in three different scenarios as a solution to the problem of information matching. Therefore, the choice of an ad-hoc retrieval task with no relevance information is purely for the purpose of showing the applicability of the IMF in a particular scenario where there is no interaction information available, and the only information available is the description of information objects (documents and queries). For the evaluation of the IMF-based ranking functions on collaborative filtering, group recommendation and ad-hoc retrieval tasks, we compare with the techniques that pervade the literature on the field of collaborative filtering and information retrieval, namely neighbourhood approaches (Section 2.2.1), latent factor models (Section 2.2.2) for recommendation, and BM25 (Section 2.1.3) and Language Models (Section 2.1.4) for ad-hoc retrieval.

The reasons for choosing these models/frameworks for comparative evaluation are:

- A majority of the successful collaborative filtering models follow either of two approaches in solving the problem of relevance matching between a user-item pair : (a) represent both user and item in a common hidden feature space and compare the relevance/similarity between them in the common hidden space (Section 2.2.1), (b) estimate the relevance of an item to a given user as the relevance of the item to the closest neighbour of the user or closest neighbour of item relevance to the user (Section 2.2.2). Therefore, we choose models from both frameworks to compare with the

IMF based ranking function applied to the problem of the CF.

- A majority of the successful probabilistic relevance ranking frameworks/functions in ad-hoc retrieval compute the probability of relevance as a matching function between the features emanating from the common vocabulary. Therefore, we choose two of the most successful models in ad-hoc retrieval, BM25 and Language Models, for evaluation.

- The IMF is applied to both collaborative filtering and ad-hoc retrieval tasks without : (a) assuming a common feature space between user-item pairs, and (b) computing explicit similarity between the users or items. However, in large scale web search engines, it is becoming increasingly common to use learning-to-rank algorithms/frameworks such as *Lambda Rank* [BRL06] for relevance ranking. However, in learning-to-rank models the results were optimized to general users and personalized ranking is difficult [SCTB$^+$12]. Moreover, as required in a solution for information matching problem, it is difficult to modify the document representation or query representation based on the interaction information due to the lack of a notion of individual document or query as the model learning is performed using the features extracted from the query-document pairs. Therefore, we choose to compare the performance with the other fundamental traditional probabilistic frameworks.

In the reminder of this chapter we present the experimental results for the ranking functions developed in Chapter 4 for various scenarios, in the order of collaborative filtering, group recommendation and ad-hoc retrieval task. We start by describing the evaluation metrics used in our experiments. We then present the experiments in each section in the following order: (a) datasets, (b) methodology, and (c) experimental results.

5.1 Metrics

We use the following standard ranking evaluation metrics for evaluating the ranked retrieval results for various tasks, namely: (a) Precision, (b) Mean Average Precision (MAP), (c) Normalised Discounted Cumulative Gain (nDCG), and (d) Mean Reciprocal Rank (MRR) .

Precision:

The precision at rank position K for the ranked list of items of length K as defined in the following way:

$$P@K = \frac{rel_K}{K} \tag{5.1}$$

where rel_K is the number of relevant items in the ranked list. In an individual recommendation retrieval scenario, the relevant items for a given user are the items with a user rating greater than the relevance threshold rating in the ranked list. The precision metric is a set-based metric and evaluates how well a model performs in putting relevant items in a top-K recommendation list, regardless of rank.

Mean Average Precision (MAP):

The MAP is defined as the average of the average precision of the ranked list (produced by the system) for a set of queries/users. The average precision (AP) of a ranked list for a given user/query is the average of the precision at each relevant item/document position in the ranked list.

The AP at rank position K is computed as follows:

$$ap@K = \frac{1}{|\mathbf{R}|}\sum_{j=1}^{K} P(K) \ . \ r_K$$

where $P(K)$ is the precision at rank position K, r_K is the binary relevance value and $\mathbf{R}$ is the total number of relevant items/documents in the collection. Now, MAP is calculated as:

$$MAP = \frac{1}{|\mathbf{Q}|}\sum_{q=1}^{|\mathbf{Q}|} ap@K$$

where $|\mathbf{Q}|$ is the number of queries/users and $ap@K$ is the average precision (AP) for the query $\mathbf{q}$ (or user $\mathbf{u}$) at rank position K. We use MAP measure for evaluating the performance on individual recommendation, group recommendation and ad-hoc retrieval tasks.

Mean Reciprocal Rank (MRR)

We also compute the performance evaluation metric Mean Reciprocal Rank (MRR) for evaluating the ranking functions on the ad-hoc retrieval task. The MRR system measure is computed as the average reciprocal rank over a set of queries. The reciprocal rank of a given query is the multiplicative inverse of the first relevant rank position item in the ranked list of items. The MRR is computed as follows:

$$MRR = \frac{1}{|\mathbf{Q}|} \cdot \sum_{q=1}^{|\mathbf{Q}|} \frac{1}{rel_q}$$

where rel_q is the rank position of the first relevant document for the query q.

In evaluating the individual and group recommendations tasks, we have choose Precision and nDCG at rank position 5 as the evaluation metrics consistent with existing literature [BCC11]. When computing Precision, we consider all the items rated by a user as 5 (in the [1-5] ratings scale) as relevant to the user, and the remaining items with user ratings and the items that he or she never rated as non-relevant to the user. When computing the nDCG score, we use the ground truth ratings as the graded relevance score of an item at the rank position. This approach is similar to the TestItems methodology used in [BCC11].

Normalised Discounted Cumulative Gain (nDCG)

We use a measure called the Normalised Discounted Cumulative Gain (nDCG) for evaluating the individual recommendation task [HKTR04]. This metric measures the quality of the ranked list by considering the ratings for the list's items with the discounted cumulative gain (DCG). We denote the rating that user $\mathbf{u}$ gives item $\mathbf{v}$ as $r(\mathbf{u}, \mathbf{v})$. The discounted cumulative gain for user $\mathbf{u}$ at rank K, or $DCG_k^{\mathbf{u}}$, is computed as:

$$DCG_K^{\mathbf{u}} = r(\mathbf{u}, \mathbf{v}_1) + \sum_{j=2}^{N} \frac{r(\mathbf{u}, \mathbf{v}_j)}{log_2(j)} \qquad (5.2)$$

We denote by $IDCG_K^{\mathbf{u}}$ the maximum gain value for the user that is obtained with the optimal re-ordering of the K items. We use this value to produce the normalised discounted cumulative gain,

Dataset	Users	Movies	Ratings	Rating scale	Label
MovieLens (100K)	983	1682	100,000	[1-5]	ML100K
MovieLens (1 Million)	6,040	3,952	1,000,000	[1-5]	ML1m
MoviePilot (Training)	171,670	23,974	4,391,822	[0-100]	MPtr
MoviePilot (Evaluation)	594	811	4,482	[0-100]	MPEval

Table 5.2: **Dataset Descriptions**. The number of users, items, and ratings as well as the two different rating scales used in each dataset. The "Label" column denotes how the algorithms are referenced in the text and other tables.

$nDCG_K^{\mathbf{u}}$:

$$nDCG_K^{\mathbf{u}} = \frac{DCG_K^{\mathbf{u}}}{IDCG_K^{\mathbf{u}}} \tag{5.3}$$

In computing the nDCG score for each user, we assign a "0" rating to all the items that were not rated by the user, so that if the model recommends any item that was not rated by the user it will be penalised.

Group Recommendation Metrics

To evaluate the quality of the group recommendation we use group precision at rank position mp, grP@mp for the group recommendation list and the Mean Average Precision (MAP) for the group, where mp is the number of items recommended to each household as defined in the CAMRa data set. The group average precision is simply the MAP computed using the set of groups instead of individual users. However, the only difference is that the precisions ($P@K$) used in computing the AP is the average of the precision over the group (household) members. A group MAP equivalent of MAP, $gMAP$, at a rank position mp as defined as follows:

$$gMAP = \frac{1}{|\mathbf{H}|}\sum_{j=1}^{|\mathbf{H}|}\frac{1}{|\mathbf{R}|}\sum_{l=1}^{mp}\text{grP}(l) \tag{5.4}$$

where $\mathbf{H}$ is a set of households, $\mathbf{R}$ is the set of relevant items for the household j, and grP(mp) is computed as the average of the precisions of individual members of the household $\mathbf{H}_j$.[1]

To evaluate the global performance, we compute the average value of each metric over all the users. We denote the overall precision and nDCG at K with $P@K$, $nDCG@K$ and use the name *information matching model* (IMM) for the ranking functions derived from the IMF.

5.2 Collaborative Filtering and Group Recommendation

First, in Section 5.2.1, we introduce the datasets that we use for collaborative filtering and group recommendation: a MoviePilot dataset of households' ratings of movies for group recommendation, and the widely used MovieLens datasets. These allow us to provide results that are comparable with the literature. Section 5.2.2 describes the methodology and metrics that we use for recommendation tasks. Finally, Section 5.2.3 details all the results we obtained.

[1]http://2011.camrachallenge.com/evaluation/

5.2.1 Datasets: MoviePilot and MovieLens

We use three different movie-rating datasets: the two publicly available MovieLens datasets (which differ in size) [MAL+03], and a MoviePilot dataset that was released as part of the Context-Aware Movie Recommendation 2011 Challenge at ACM RecSys [Sai11], or CAMRa. We note that the main task of the first track of the challenge was to address the same problem[2] that we describe here: that is, recommending a given (mp) set of items to a household of users.

The characteristics of the datasets are given in Table 5.2: there is a wide range in both the number of users and movies, and the MoviePilot data uses a 0 to 100 rating scale (as opposed to MovieLens' 1-5 star ratings). Furthermore, the MoviePilot data contains 290 unique households with between two to four members each: the majority of the user-ids have been assigned to a household. We note that, although previous research has used the MovieLens data to examine group recommendation scenarios, these datasets do not include any explicit group membership data. In the past, researchers have overcome this by forming implicit groups in the data [BMR10]. However, since the MoviePilot data does contain group membership features, we solely use the MovieLens data to provide results for recommendations to individuals that allow our model to be compared with the broader CF literature.

5.2.2 Methodology

It is common to design and evaluate recommendation systems around the notion of predicting individual ratings. In other words, a good system is the one which predicts the numerical 1–5 rating most accurately. However, such evaluation is away from our true task, which is to make recommendations to a user. Following other authors, such as [CKT10], we prefer to evaluate the ranking of recommendations offered to a user – this of course fits more closely with the traditional IR form of evaluation. Thus, our main objective in the evaluation is to test how well the IMF relevance ranking function places the most relevant set of items at the top of the recommended list of items.

Given the datasets above, we now describe how we test and measure the efficacy of the recommendation algorithms described in Sections 2.2.1, 2.2.2 and our IMM. We tested the algorithms in two settings: how well they can produce recommendations for individuals, and how well they produce recommendations for groups. We then compare these two settings to examine how much performance is lost between the two scenarios.

Recommendations to Individuals. First, we verify how our model performs when used to compute recommendations for *individuals*, compared to the other approaches. The main purpose for doing so is two-fold: (a) in previous chapters we posited that better group recommendations could be obtained by more accurate models of group members, and (b) we demonstrate that we compare against strong baselines by reproducing results from [BCC11].

In our evaluation, we rank *all* the items that each target user has not rated in the training set: for ML100K, ML1M and MoviePilot we ranked more than 1500, 3800 and 23000 items respectively. We then set those test items that the user rated higher than a given threshold (e.g., 4 stars or higher on the 1-5 scale) as relevant and all other items - both those rated below the threshold or not rated at all, as not

[2]http://2011.camrachallenge.com/call-for-papers/

relevant. We note that, in doing so, we will tend to observe very low performance scores. We rank all the items that were not rated by the user for two reasons: (1) it is closer to the task that practical systems must perform, and (2) it produces results with a more accurate distinction between different models, in terms of performance.

To conduct our experiments with the MovieLens data, we have randomly divided the datasets into training (60% of the data) and test sets, making sure that ratings from any given user are in both training and testing. We repeat this process five times to compute 5-fold cross validated results. We also used the MoviePilot data, by disregarding the group memberships. We adopted the Apache Mahout library implementation for the user-based method , the item-based method with the Pearson correlation similarity and PureSVD method based on matrix factorsiation. We denote the results obtained by these methods $k-$User, k-Item and PureSVD.

Recommendations to Groups. Next, we experiment with the extent to which the algorithms can produce quality recommendations for *groups*, using the MoviePilot data. The methodology that we adopt seek to align itself to the structure of the CAMRa challenge. The MoviePilot data contains a set of households H already divided into training and evaluation sets: the task of the challenge was to recommend a specified number of items to each household; data splitting and the specification number of items to recommend to each household was defined by the organisers; the range number of items recommended to each household is between 1 and 82. We thus use the training set (MPtr) to learn the model and rank all the non-rated items for each user and compare to the evaluation set (MPEval) for both the recommendation tasks.

5.2.3 Experimental Results

Since we will be providing both reference (individual recommendation) and group recommendation results, in the following we describe each evaluation strategy separately. Later, we compare these two sets of experiments to examine how group and individual recommendation contexts differ.

5.2.4 Recommendations for Individuals

In order to rank a set of items in the order of relevance for a given user $\mathbf{u}_m$, we compute the probability of relevance between $\mathbf{u}_m$ and every item $\mathbf{v}_n$ in item set $\mathcal{V}$, and use the CF ranking function from the Chapter 4 (page 91). For the ease of reading, we present the CF ranking function from Chapter 4.

CF Ranking function:

$$
\begin{aligned}
p(r=1|\mathbf{u}_m,\mathbf{v}_n) \propto_r & \sum_{\text{(k, l)}:r_{kl}\in\mathcal{D}^r} log\left\{\frac{p(rt_{mk}|\alpha_k=1)}{\sum_{\alpha_k\in\{0,1\}} p(rt_{mk}|\alpha_k)p(\alpha_k)} \cdot \frac{p(rt_{nl}|\beta_l=1)p(\beta_l=1)}{\sum_{\beta_l\in\{0,1\}} p(rt_{ln}|\beta_l)}\right\} \\
\propto_r & \sum_{\text{(k, l)}:r_{kl}\in\mathcal{D}^r} log\left\{\frac{e^{-\mu_k(1)}\mu_k(1)^{r_{kl}}}{p_i e^{-\mu_k(1)}\mu_k(1)^{r_{kl}} + (1-p_k)e^{-\mu_k(0)}\mu_k(0)^{r_{kl}}}\right\} \\
& + log\left\{\frac{e^{-\mu_l(1)}\mu_l(1)^{r_{kl}}}{p_l e^{-\mu_l(1)}\mu_l(1)^{r_{kl}} + (1-p_l)e^{-\mu_l(0)}\mu_l(0)^{r_{kl}}}\right\}
\end{aligned}
$$

In order to apply IMM, we need to define a set of relevant pairs and learn the 2-Poisson mixture parameters.

To apply the information matching model relevance ranking function, we need to define the relevant set of user-item pairs, i.e $\mathcal{D}^r$, and learn 2-Poisson mixture parameters. Therefore, we define a set of relevant pairs by taking the user-item pairs in the training set with a rating greater than or equal to 3. To learn the 2-Poisson parameter values for each user and item, we use Expectation Maximization [DLR77] along with the ratings received by user and item respectively as described before. An additional assumption made in learning the parameter values is: we assume that the user's non-rated items were not relevant to the user and assign the lowest possible rating. To effectively learn the difference in 2-Poisson means from the ratings of a user, we assigned a rating value "0" to non-rated items as opposed to "1".

Table 5.3 summarises the performance of each model on different collections, with respect to each evaluation metric, where an item is considered to be relevant if the rating given by the user is 5 for MovieLens and greater than 70 for the MoviePilot data. Most notably, we have only reported MAP scores for the MoviePilot data. The reason for this is that the performance of the neighbourhood and latent factor models was close to 0[3]. Furthermore, the simple popularity[4] based recommendation often produces better results than the personalised alternatives. However, we clearly observe that the IMM performs better than the all the remaining models on all the metrics. We also note that the IMM performance on P@10, P@20, nDCG@10, nDCG@20 is also higher when compared with the other models.

We also test the extent that performance on individual recommendation varies when we change the relevance threshold from 3 to 5 (MovieLens) or 70 to 100 (MoviePilot). Figure 5.1 shows the performance of both PureSVD, and the IMM across these thresholds: the precision increases as the relevance threshold rating is reduced, but the IMM continues to show better performance than the PureSVD approach.

Another evaluation question which arises for recommendation systems is the effectiveness of ranking items solely on popularity. In contrast to IR, this is in fact an effective procedure as demonstrated in [CKT10]. However, it is often ignored in the recommendation system literature, due to the fact that it does not provide a personalised form of recommendation. To have a fair comparison, we therefore examine the popularity effect, and evaluate our model accordingly. For a non-personalised recommender, we use a simple popularity based recommendation model denoted as *Pop-item*. *Pop-item* recommends a fixed set of top-K ranked items to any user, and these top-K items are computed based on the number of ratings received from users. From Table 5.3, we can clearly see that the IMM outperformed the popularity based one for the precision measure, while other methods such as PureSVD failed.

An issue in the CF datasets is that the majority of the user ratings are given on the most popular items in the collection (approximately 40% of the 1M dataset ratings are on 6% of the movies [CKT10]). In general, the small set of popular items are referred to as *short-head* and the remaining less popular items as the *long-tail*. As popular items can easily be identified by the user, a good recommender system should also recommend non-trivial items from the *long-tail*.

To test the ability of our model to recommend non-trivial items, we evaluate it on the *long-tail* items

[3]A similar result was reported on the competition web page: `http://2011.camrachallenge.com/2011/11/final-evaluation/`

[4]Popularity is simply the number of observations received from the users.

		Algorithm				
Collection	Metric	k-Item	k-User	PureSVD	*Pop-item*	IMM
ML100K	P@5	0.00135	0.006	0.067	0.227	0.267
	NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
	MAP	0.013	0.041	0.061	0.119	0.156
ML1m	P@5	0.00047	0.071	0.093	0.149	0.175
	nDCG@5	0.0009	0.094	0.113	0.233	0.234
	MAP	0.00677	0.0320	0.063	0.109	0.128
MPEval	MAP	0.00061	0.00238	0.004	0.12	0.208

Table 5.3: **Recommending to Individuals**. Performance results, in terms of Precision@5, Non-Discounted Cumulative Gain@5 and MAP, when using the popularity baseline, neighbourhood approaches (item and user based), PureSVD, and the Information Matching Model (IMM) in order to generate recommendations for individual users.

		Model & Performance			
Collection	Metric	*Pop-item*	PureSVD	IMM	% change
M100K	P@5	0.0266	0.0217	**0.0376**	41.3
	nDCG@5	0.0384	0.0473	**0.071**	50.1

Table 5.4: The performance comparison of the IMM with the *pureSVD* and the *Pop-item* on long-tail recommendation.

of the M100K dataset. To obtain the *long-tail* items, we set up a threshold and remove popular items so that 30% of the ratings are removed. We repeat the same experiments but using only the *long-tail* items. Table 5.4 summarises the results of the best three models on *long-tail* collection. From the table, we can see that the IMM largely outperforms the pureSVD method, despite the fact the the performance of the PureSVD increases in this case and bears comparison with the popularity-based recommender. We believe of the performance gain is due to the fact that our model makes use of all the related ratings without resolving to a lower dimension, which would inevitably reduce the accuracy.

In conclusion, the individual recommendation results show that the two best performing personalised recommender models are the PureSVD and the IMM. In the following section, we show how similar results emerge in the group recommendation scenario.

5.2.5 Recommendations for Groups

We evaluate group recommendation with our models when using both of the previously described strategies: (a) by creating a profile for each group based on merging all members' ratings, and (b) by merging members' ranked lists of items using the *Least Misery* strategy. As before, we also repeat the experiments for various relevance rating thresholds.

Table 5.5 summarises the performance of each model; we exclude the results from both the kNN

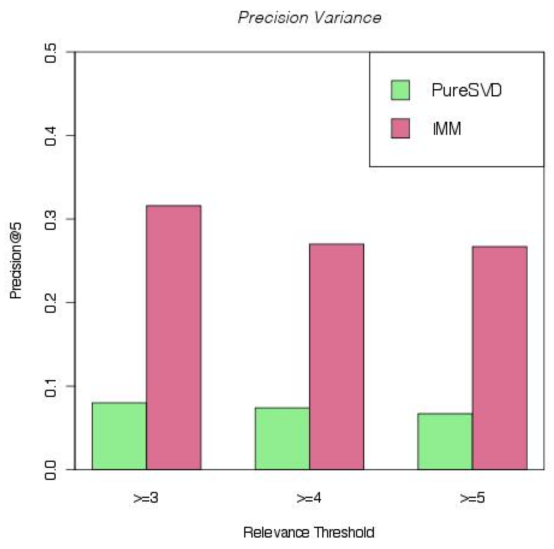


Figure 5.1: **Recommending to Individuals**. Precision@5 variance in performance, when using PureSVD and the IMM on the MovieLens 100k dataset, as the threshold of relevance is changed.

user- and item-based approaches as they were all zeroes. Once again, it is clear that the group recommendation model based on the IMM outperforms the other two methods.

We note that the MoviePilot data does not contain the group information for all the users in the training data. So, when we merge the group profiles, the items considered in training were the items rated by at least one member who has a group identifier. As a result, we lack the rating information of the items that were not rated by any member of any group, and the total number of items ranked for each user is less than the total number of potential test items of 23000 (around 7710 item) as compared to the other two methods. From Table 5.5, we can observe that the performance of the profile aggregation model is higher than the performance obtained for the merging ranked lists model. However, the group grP@mp (mp is the ranking position given by MoviePilot as part of the challenge) score is higher for the merging ranked lists model than the aggregated user profile.

5.2.6 Performance Loss Between Individual and Group Recommendation

Finally, we measure how much recommendation quality degrades between the individual and group scenarios. Figure 5.2 (page 104) illustrates the relative performance of individual and group recommendation of the PureSVD and the IMM (in terms of P@5), as different relevance rating thresholds are tested.

It is clear that, throughout all approaches, a loss is incurred when transitioning from individual to group recommendations. This captures the difficulty of the group recommendation problem, and may be explained by the number of different assumptions that each algorithm must make. However, not only does the IMM obtain better results compared to the PureSVD, it also degrades less when used for group recommendation. In fact, precision wanes by up to 26% when IMM is used for group recommendation; the PureSVD approach, instead, loses between 50% and 95% of the precision it had for individuals when applied to groups.

		Merging Individuals' Profiles		
Model	Metric	≥ 70	≥ 80	≥ 90
PureSVD	grP@mp	0.00027	0.00013	0.000034
	gMAP	0.00416	0.00345	0.00221
		Rank Aggregation By Least Misery		
	Metric	≥ 70	≥ 80	≥ 90
PureSVD	grP@mp	0.0015	0.00071	0.0001
	gMAP	0.0029	0.0021	0.0017
		IMM based Least Misery		
	Metric	≥ 70	≥ 80	≥ 90
IMM	grP@mp	0.142	0.112	0.068
	gMAP	0.159	0.147	0.120

Table 5.5: **Group Recommendation.** Performance results on the MoviePilot data for varied relevance rating thresholds. We include group precision (grP@mp) and Group Mean Average Precision (gMAP) for (a) PureSVD on group profiles that are merged ratings of group members, (b) individual PureSVD results that were merged with *Least Misery*, and (c) the group information matching model.

5.3 Ad-hoc retrieval

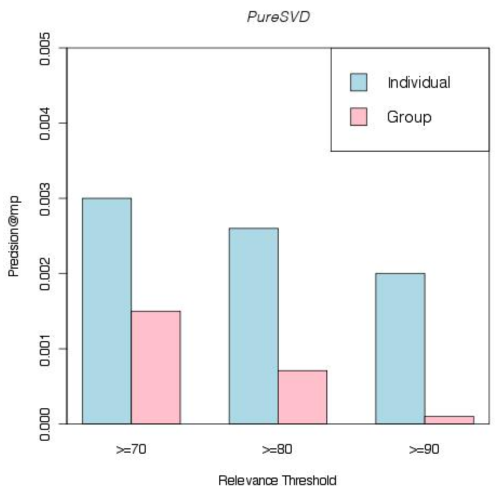
The main objective of our experiments for the ad-hoc retrieval task is to test the performance of the IMM based relevance ranking function when it uses exactly the same information from the documents and the query (i.e. term frequency statistics) as the other strong baseline ranking functions in ad-hoc retrieval.

5.3.1 Datasets: TREC collections

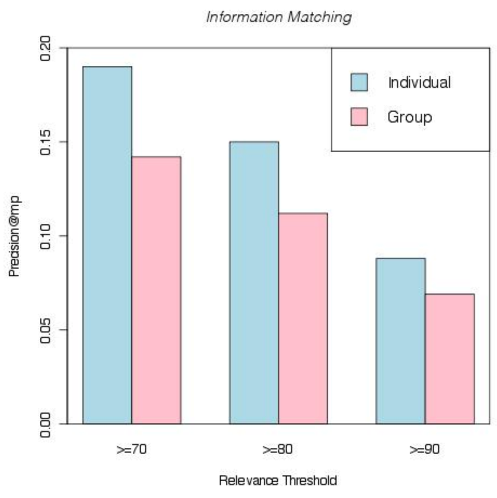
For the evaluation, we experiment with the following five TREC collections representing small to medium sizes: 1) FBIS on disc 5, 2) Financial Times (FT) on disk 4, 3) Los Angeles Times (LA) on disk 5, 4) TREC-7 and TREC-8 ad-hoc retrieval document collection, Disk 4 & 5 minus Congressional Record, and 5) WT10G collection. The topic sets used are: 1) topics 301-350, 2) topics 401-450, 3) topics 501-550 and 4) topics 301-350 and 601-700 minus 672. We use the document collection followed by the TREC number as a label for the test collection, e.g. FBIS-8 represents the test collection with the FBIS document collection and TREC-8 topics (i.e. 401-450). Similarly, label *Robust* and *TREC-10* represent the TREC 4&5 document collection with Robust topics, and WT10G collection with topics 501-550 respectively. The test collections and their corresponding labels used in the evaluation are shown in Table 5.6.

5.3.2 Methodology

For the performance evaluation, we follow the standard method of evaluation of ad-hoc retrieval ranking functions, i.e., evaluating based on the ranking evaluation metric. Specifically, we computed Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) metric values using the top 1000 retrieved documents and compare them with the popular retrieval models, BM25 [RZ09] and Language Models



(a) PureSVD



(b) IMM

Figure 5.2: **Performance Loss Between Individual and Group Recommendations**. We plot, for PureSVD and IMM, the performance change as the relevance threshold is varied. Note that the rating scales for each plot are different, and that the PureSVD consistently has a higher loss in performance when used for group recommendation (compared to when it is used for individual recommendation).

[PC98] (with the Jelinek-Mercer smoothing and Dirichlet prior). The baseline models, labels and ranking functions associated with each model are described in Table 5.7 and Table 5.8 respectively.

Before employing each baseline model on the test collections, we preprocess documents and queries by removing stop-words and by applying the Porter stemmer. Furthermore, in order to compute the best performance score for each model, we tune the respective parameters for each model, i.e., (b, k) in BM25, and λ and μ in the Language model with JM and the Dirichlet prior respectively. Our baseline scores are consistent with others, such as the ones reported in [ZL04] and by Yahoo[5].

There are two main aspects of the implementation of the IMM ranking function:

1. The document length normalisation: an issue with the Poisson mixture assumption is that it assumes a fixed document length for all documents in the collection [Har75a, RW94]. So, we need to normalise the document length before we apply our model based on 2-Poisson mixture.

2. Parameter initialisation for learning the mixture parameters.

Document length:

The 2-Poisson model assumption of a fixed document length for all documents in the collection is not a valid assumption as the document length varies in the collection. In order to circumvent this problem, two hypotheses are proposed to explain the varied document lengths in the collection, namely Scope Hypothesis and Verbosity hypothesis and these are incorporated into various models [RW94, Sin97]. For our experiments on the IMM evaluation, we assume the verbosity hypothesis, i.e. lengthy documents

[5]`http://barcelona.research.yahoo.net/dokuwiki/doku.php?id=baselines`

Collection	301-350 topics	401-450 topics
FBIS	FBIS-7	FBIS-8
FT	FT-7	FT-8
LA	LA-7	LA-8
TREC 4 & 5	TREC-7	TREC-8

(a) collection one

Collection	501-550 topics	Robust-2004 topics
WT10G	TREC-10	-
TREC 4 & 5	-	TREC-Robust

(b) collection two

Table 5.6: TREC collections and the labels used in Ad-hoc retrieval experiments.

Model Name	Label
Information Matching Model with EM	IMM-EM
Information Matching Model with Gibbs sampling	IMM-Bayesian
Best Match-25	BM25
Language Model with Jelinek-Mercer smoothing	LM-JM
Language Model with Dirichlet prior	LM-Dirichlet

Table 5.7: The lables used for representing Ad-hoc retrieval models.

cover the same scope of short document but use more words, and incorporate it into our ranking function by modifying the term frequency as,

$$tf = \frac{\hat{tf}}{\left(1 - b + b\frac{DL}{avgDL}\right)}, \tag{5.5}$$

where $avgDL$ is the average document length in the collection, DL is the document length, $b \in [0, 1]$ and $\hat{tf}$ is the observed term frequency. The normalisation formula in Eq. (5.5) penalises the term frequency in lengthy documents based on the preset b value.

Parameter initialisation:

The second important issue in the IMM configuration is to initialise the mixture parameters. The initial choice of the parameter values greatly affects the learning in EM and the burn-in period of the Gibbs sampling. In general, the parameters can either be initialised randomly or initialized based on collection statistics information. For all of our experiments, we initialise the parameters with the help of data collection statistics.

For the EM algorithm, we initialise each p as the portion of the document collection with the presence of term, i, in the document. Thus the initial value is same as the IDF weighting (as discussed previously). We used a minuscule value to initialise $\mu(0)$ by assuming that the average term frequency of a term associated with the term-description elite property in a document approaches zero if it is non-elite

Model	This is the first $S(\mathbf{q},\mathbf{d})$: Ranking score for $(\mathbf{q},\mathbf{d})$ pair where the query $\mathbf{q} = \langle q_1, q_2, \cdots, q_{\lvert\mathbf{q}\rvert} \rangle$ and $q_i \in \mathbf{q}$	Parameters
BM25	$\sum_{q_i \in \mathbf{q}} \frac{f(q_i,\mathbf{d})(k_1+1)}{k_1[1-b+b\frac{dl}{avgDL}]+f(q_i,\mathbf{d})} \left(log \frac{(N-n(q_i)+0.5)}{(n(q_i)+0.5)} \right)$	$f(q_i,\mathbf{d})$ is the q_i term frequency in $\mathbf{d}$, $n(q_i)$ is the number of documents containing q_i and N the is total number of documents. We set $k_1 = 1.2$ and $b = 0.75$.
LM-JM	$\prod_{q_i \in \mathbf{q}} (1-\lambda) \frac{f(q_i,\mathbf{d})}{\lvert\mathbf{d}\rvert} + \lambda\, p(q_i \vert C)$	$p(q_i \vert C)$ is the background probability and we set $\lambda = 0.1$.
LM-Dirichlet	$\prod_{q_i \in \mathbf{q}} \frac{f(q_i,\mathbf{d}) + \mu p(q_i \vert C)}{\lvert\mathbf{d}\rvert + \mu}$	μ varies for each collection and its values are set between 1500 and 2000.
IMM	$\sum_{q_i \in \mathbf{q}} log \left(\frac{e^{-\mu_i(1)} \mu_i(1)^{f(q_i,\mathbf{d})}}{p_i e^{-\mu_i(1)} \mu_i(1)^{f(q_i,\mathbf{d})} + (1-p_i) e^{-\mu_i(0)} \mu_i(0)^{f(q_i,\mathbf{d})}} \right)$	All 2-Poisson mixture parameters μ and p_i are estimated.

Table 5.8: A summary of Ad-hoc retrieval relevance ranking functions used in experiments.

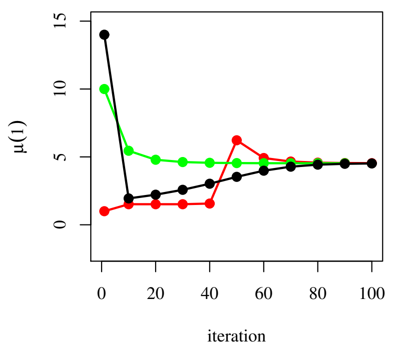
to the document. Similarly, $\mu(1)$ is initialised with the average number of times that the term appears in the document collection with its term frequency in a document being greater than one.

The initial choice of parameters greatly affects the burn-in period of the Bayesian sampling. So, similarly, we choose the prior parameters values based on the collection statistics. The initial value of p is set to be the portion of the documents collection with the term (the same as the assumption used to show the ranking function relationship with IDF), and $\mu(1)$ is initialised with the average number of times that the term appears in the document with its term frequency being greater than one. In the following sections we describe both of these aspects.

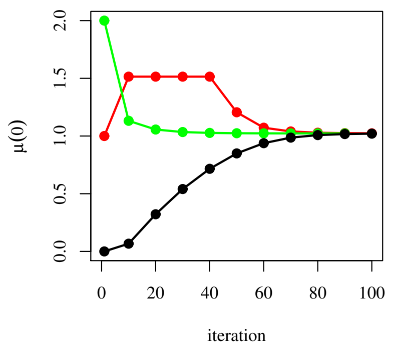
5.3.3 Learning the parameters

To test the stability of our learning algorithms, we have initialised the parameters with different values and run the learning process. In all the cases, EM converges to a stable point. For example, as shown in Fig. 5.3, $\mu(0)$, $\mu(1)$ and p are converged to 4.53, 1.004 and 0.14, respectively, for the term "time" in the TREC 4 & 5 disk collection. The document collection has 360514 documents (close to 70% of the documents) with the term "time". As shown in the Figure 5.3, the converged value of $\mu(0)$ (representing the average number of times that a term can occur in the non-elite set of documents in the collection) is close to 1 following the collection statistics. The same case applies for $\mu(1)$ as well, which is close to 4.53.

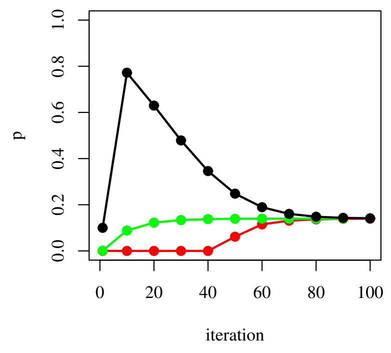
To study the effect of the document length normalisation parameter b on the performance of the



(a) The convergence of the parameter $\mu(1)$ for three different initialisations.



(b) The convergence of the parameter $\mu(0)$ for three different initialisations.



(c) The convergence of the mixture probability parameter p.

Figure 5.3: The EM convergence for the term "time" in the TREC 4 & 5 collection for different initialisations.

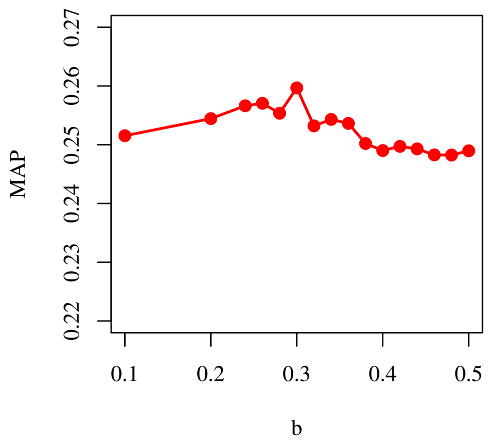


Figure 5.4: The impact of the parameter b over the Mean Average Precision on TREC-8.

ranking function, we fixed the b value to 0.3 for learning the mixture model parameters. Then we ran the retrieval evaluation experiments by varying the b value. The graph in Fig. 5.4 shows that the MAP score peaks at 0.3 and the score is always more than 0.25, which is more than MAP score for the default parameters settings of BM25 and the two language models. The effect of b on MAP is similar for all other collections. This demonstrates that the document length normalisation is helpful in the model.

We also initialise the mixture parameters by using the collection statistics as follows. For the EM algorithm, we initialise p as the percentage of the documents where the term occurs. Thus the initial rank function is equivalent to the IDF weighting (see the discussion in Section 3). We use a minuscule value to initialise $\mu(0)$ by assuming that the average term frequency of a term associated with the term-description elite property in a document approaches zero if it is non-elite to the document. Similarly, $\mu(1)$ is initialised with the average number of times that the term appeared in document collection with its term frequency in a document being greater than one. For the Gibbs sampling, we choose the prior parameters values in the similar fashion.

To circumvent the document length problem, we assume the verbosity hypothesis [RW94] and incorporate it into our ranking function by normalising document length similar to BM25 [RW94]. For all the experiments, we fix the document length normalisation parameter. Note that one can further optimise it by applying the method proposed in [LZ11].

5.3.4 Experimental results

After learning the parameters from each document collection, we employ our ranking function in Eq. (4.31) (described in Table 5.8 and page 78) on each test collection in Table 5.6. Table 5.9 summarises the performance of the IMM ranking function (IMM), using EM & Bayesian estimation, along with the performance of other influential ranking methods. The results from each model are labelled with the labels shown in Table 5.7. For the IMM, we fix the b value for the training as 0.3 and conduct experiments by varying its value. The graph in Fig. 5.4 shows the MAP score for different values of b on

		Model Name & Performance				
Collection	Metric	BM25	LM - JM	Dirichlet-LM	IMM-EM	IMM-Bayesian
FT-8	MAP	0.323	0.317	0.325	0.347	**0.347**†
	MRR	0.649	0.590	0.664	0.711	**0.724**†
FBIS-8	MAP	0.326	0.306	0.325	0.315	**0.334**†
	MRR	0.598	0.496	0.598	0.560	**0.614**†
LA-8	MAP	0.254	0.232	0.256	0.260	**0.276**†
	MRR	0.565	0.402	0.545	0.583	**0.594**†
TREC-8	MAP	0.251	0.239	0.256	0.257	**0.260**
	MRR	0.644	0.476	0.638	0.654	**0.670**†
TREC-7	MAP	0.193	0.180	0.192	0.191	**0.195**
	MRR	0.652	0.551	0.650	0.630	**0.667**
Robust	MAP	0.242	0.185	0.245	0.245	**0.248**
	MRR	0.650	0.564	**0.668**	0.620	0.638
TREC-10	MAP	0.193	0.148	0.193	0.190	**0.195**
	MRR	0.596	0.451	0.588	0.60	**0.611**

Table 5.9: The performance comparison of the IMM with other baseline models. $t\text{-}test$ with 95% confidence is used and the statistically significant results (with respect to the second best model) are marked with †.

TREC-8. We also find that the MAP and the MRR cannot be optimised simultaneously and this confirms the theoretical argument about the trade-off between MAP and MRR in [WZ10].

From Table 5.9, we can see that our ranking function outperforms other models in most cases (some of them are statistically significant). Because the rank function does not use any information other than the term statistics in the document collection, we believe that the improvement is due to the term-based parameter estimation, similar to the per-term smoothing in the Poisson based query-generation language models [MFZ07]. Moreover, the performance of our model on title queries is comparable to the improved reported results in [MFZ07]. In summary, the ad-hoc retrieval experiments show that the IMF has great potential in text retrieval. A simple ranking function derived from our new theory of information matching demonstrates that it can handle the retrieval situation *without* relevance feedback (or the interaction).

5.4 A preference based search engine

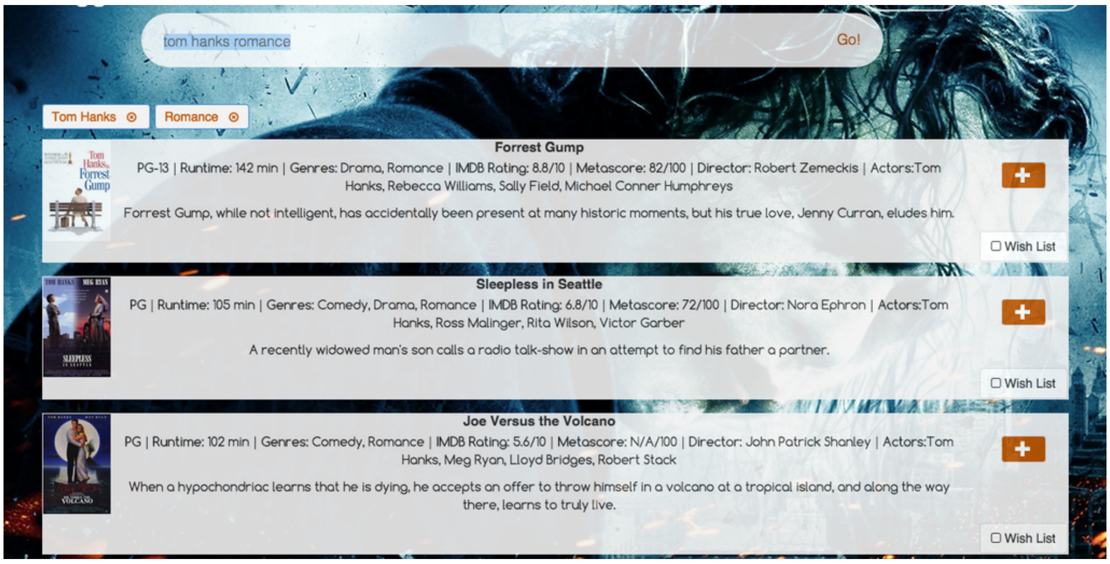
In this section we show how we can build a preference based search engine for movies using the IMF and incorporate real-time interaction while computing relevance matching. If we observe carefully, while applying the model, the IMF represents the users and movies in two distinct feature spaces, and computes the relevance match based on the learned preference mapping between these features. This allows the model to independently update the user features and movie features in their respective feature spaces in light of new information (e.g. real-time interaction). This means that any real-time information coming on a specific user or movie can be immediately used to update their respective feature representations. Similarly, the parameters in the mapping matrices are dependent on the independent interaction between the users and the movies. Therefore, the model can update mapping parameter values with new information. However, as specified earlier, these computations are expensive.

We can build a real-time preference search/matching system based on the IMF by learning a model using some past data and then by:

- updating the user feature parameters (α_k^m's of user $\mathbf{u}_m$) after gathering a certain amount of interaction information on movies. The minimum information required is a single interaction (e.g. click). And the update can be performed after every interaction.
- updating movie feature parameters (β_l^n's of movie $\mathbf{v}_n$) after gathering a certain amount of interaction information on the movie by the users. In practical systems, we could update each movie feature parameter distributions overnight based on all interactions on a particular day.
- updating the mapping parameters (parameters in $\mathbf{G}$ and $\mathbf{H}$) after gathering a set of interactions between users and movies. In practice, we could also update these parameter distribution values overnight.

Now, we can make the retrieval faster by indexing each movie with the user features that they satisfy and having their corresponding feature relevance probability computed using the preference mappings. This is in contrast to computing the relevance probability by going through the preference mappings for each user (or session) with the same feature. For example, the model could index the movie "Sherk" with the user feature corresponding to the movie "Cars', as both are animation movies. Now, for any user with the user feature "Cars', the relevance of "Shrek" based on that feature is simply the indexed probability. Using these simple observations and restricting the mapping matrices to the size of observed interactions we have built a simple preference search engine where users can search movies with their preferences represented with a movie, cast, genre or any combination of these preferences. For example, a user can search for movies with his preferences expressed as "tom hanks romance". In this particular example the user preferences are expressed though an actor name "Tom Hanks" and a genre "romance".

We first describe some example results and then describe the data that is used to build the system. The demo system can be accessed on the internet at `http://jaggu.com`. The results from the system for the above "tom hanks romance" preference query can be seen in the Figure 5.5. The results are relevant if user's preferences are expressed though "Tom Hanks" and "romance", as the top three result movies, namely Forrest Gump, Sleepless in Seattle and Joe Versus the Volcano, are romantic movies starring Tom Hanks. However, the system will also return other movies that satisfy the preferences with a different cast depending on the strength of the information with which the model is trained. For example, the results in Figure 5.6 show that with the query phrase "children of heaven" as preferences, the system retrieves other family and children oriented movies. The top result movies, "Spirited Away" and "Love Actually", are movies about the family with children. Finally, the results in Figure 5.7 are the results of searching with the preferences "romance comedy" and after clicking on the result Annie



tom hanks romance

Tom Hanks

Romance

Forrest Gump

PG-13 | Runtime: 142 min | Genres: Drama, Romance | IMDB Rating: 8.8/10 | Metascore: 82/100 | Director: Robert Zemeckis | Actors:Tom Hanks, Rebecca Williams, Sally Field, Michael Conner Humphreys

Forrest Gump, while not intelligent, has accidentally been present at many historic moments, but his true love, Jenny Curran, eludes him.

Sleepless in Seattle

PG | Runtime: 105 min | Genres: Comedy, Drama, Romance | IMDB Rating: 6.8/10 | Metascore: 72/100 | Director: Nora Ephron | Actors:Tom Hanks, Ross Malinger, Rita Wilson, Victor Garber

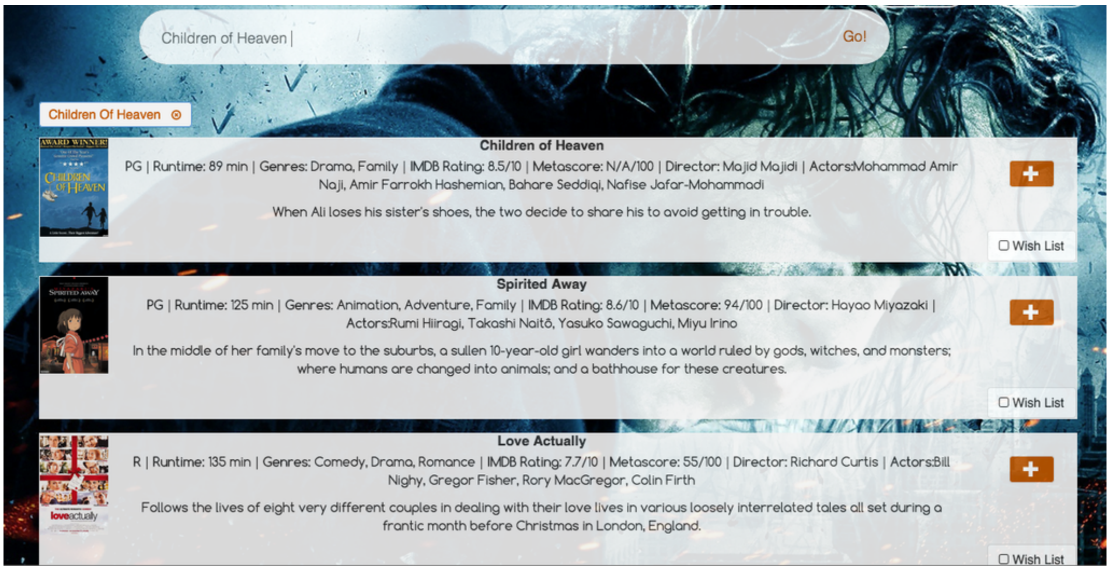
A recently widowed man's son calls a radio talk-show in an attempt to find his father a partner.

Joe Versus the Volcano

PG | Runtime: 102 min | Genres: Comedy, Romance | IMDB Rating: 5.6/10 | Metascore: N/A/100 | Director: John Patrick Shanley | Actors:Tom Hanks, Meg Ryan, Lloyd Bridges, Robert Stack

When a hypochondriac learns that he is dying, he accepts an offer to throw himself in a volcano at a tropical island, and along the way there, learns to truly live.

Figure 5.5: Search results for the preference query "Tom Hanks romance". The results are Tom Hanks's highly successful romantic movies, as relevant to the preferences of the query.



Children of Heaven

Children Of Heaven

Children of Heaven

PG | Runtime: 89 min | Genres: Drama, Family | IMDB Rating: 8.5/10 | Metascore: N/A/100 | Director: Majid Majidi | Actors:Mohammad Amir Naji, Amir Farrokh Hashemian, Bahare Seddiqi, Nafise Jafar-Mohammadi

When Ali loses his sister's shoes, the two decide to share his to avoid getting in trouble.

Spirited Away

PG | Runtime: 125 min | Genres: Animation, Adventure, Family | IMDB Rating: 8.6/10 | Metascore: 94/100 | Director: Hayao Miyazaki | Actors:Rumi Hiiragi, Takashi Naitô, Yasuko Sawaguchi, Miyu Irino

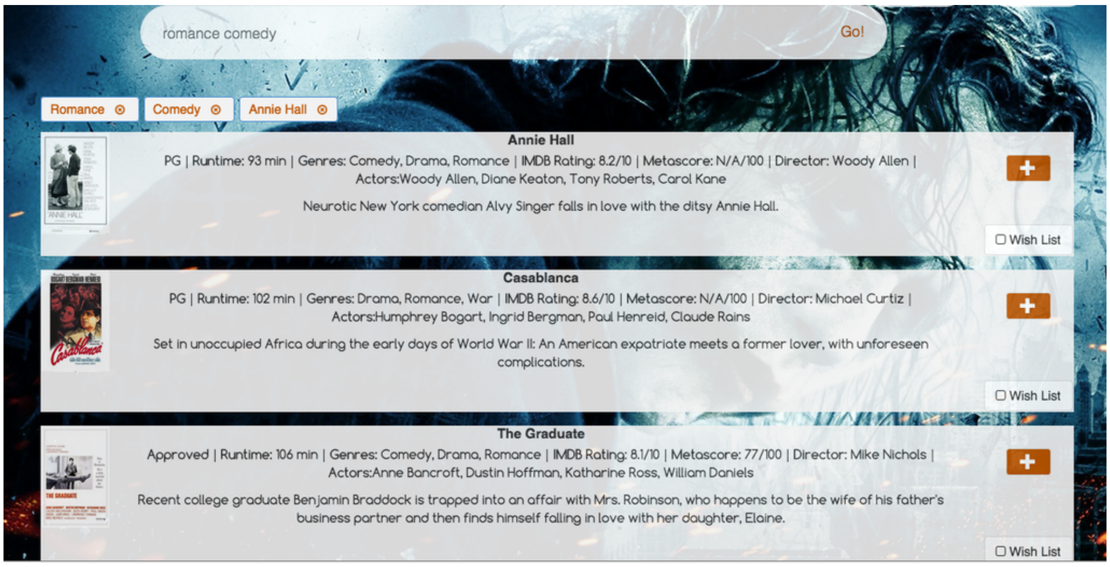
In the middle of her family's move to the suburbs, a sullen 10-year-old girl wanders into a world ruled by gods, witches, and monsters; where humans are changed into animals; and a bathhouse for these creatures.

Love Actually

R | Runtime: 135 min | Genres: Comedy, Drama, Romance | IMDB Rating: 7.7/10 | Metascore: 55/100 | Director: Richard Curtis | Actors:Bill Nighy, Gregor Fisher, Rory MacGregor, Colin Firth

Follows the lives of eight very different couples in dealing with their love lives in various loosely interrelated tales all set during a frantic month before Christmas in London, England.

Figure 5.6: Search results for the preference query "Children of Heaven". The results shows that the system retrieves other relevant family and children movies for these preferences.



Annie Hall
PG | Runtime: 93 min | Genres: Comedy, Drama, Romance | IMDB Rating: 8.2/10 | Metascore: N/A/100 | Director: Woody Allen |
Actors:Woody Allen, Diane Keaton, Tony Roberts, Carol Kane
Neurotic New York comedian Alvy Singer falls in love with the ditsy Annie Hall.

Casablanca
PG | Runtime: 102 min | Genres: Drama, Romance, War | IMDB Rating: 8.6/10 | Metascore: N/A/100 | Director: Michael Curtiz |
Actors:Humphrey Bogart, Ingrid Bergman, Paul Henreid, Claude Rains
Set in unoccupied Africa during the early days of World War II: An American expatriate meets a former lover, with unforeseen complications.

The Graduate
Approved | Runtime: 106 min | Genres: Comedy, Drama, Romance | IMDB Rating: 8.1/10 | Metascore: 77/100 | Director: Mike Nichols |
Actors:Anne Bancroft, Dustin Hoffman, Katharine Ross, William Daniels
Recent college graduate Benjamin Braddock is trapped into an affair with Mrs. Robinson, who happens to be the wife of his father's business partner and then finds himself falling in love with her daughter, Elaine.

Figure 5.7: Search results for a combination of the initial query "romance comedy" and the interaction click on the movie "Annie Hall".

Hall. Therefore, the results are a combination of the initial preferences and the real-time interaction. This system demonstrates that in computing the relevance match between user-movie pairs, the IMF combines information on the movie, user, and their past interactions (e.g., ratings), along with other users' and movies' interaction data available to the system, and produces relevant results.

5.4.1 System details

In order to build the preference based matching system we crawled the web and collected information about 27,232 English movies from around 5,61,000 users. We have also gathered 3.1 million interactions between these users and the movies. The interactions in this particular scenario were the ratings on different scales, and the scale is dependent on the source. For example, Rotten Tomatos[6] has a rating scale between 0 and 100. Some other sources have ratings between 0 and 10. For the movie specific information, we have also collected each movie's genre and the cast. Using this dataset, we represented users and movies with the features created from the observations on them, and then we trained the IMF model. For example, an uncertainty in a given user feature is represented with a feature corresponding to movie "Cars" if and only if we observe some user data with the movie "Cars". Similarly, we also restricted the movie features with the set of features containing the genre, cast and the features corresponding to the user's that had past interaction with the movie (e.g., rating). As stated in previous chapter, the computation of the model is expensive even with these restrictions due to the explosion of the number of representational features. Once the model is learned, we index each movie with the user features (movies, cast, genre, etc.) and the probabilities corresponding to that movie satisfying any user with that feature learned using the IMF. From the index, we retrieve all the movies that are indexed with the given user features and combine their score to get the final relevance ranking score used to rank movies.

The primary intention of developing this system is to show that we can build a real-time preference search matching system using the IMF. However, we end with a note that there are a number of challenges

[6]`http://www.rottentomatoes.com/`

in making the computation of the model less expensive.

5.5 Conclusion

In this chapter, we have validated the proposed IMF framework for three different retrieval scenarios by evaluating the performance of ranking functions on several publicly available test collections (The TREC collections for ad-hoc retrieval task, and MovieLens and MoviePilot for individual and group recommendation). Our objective was to gather some experiential evidence in support of using the IMF in three different retrieval scenarios. We found that the IMF ranking function performance is on par or improved in comparison with the popularity-based baseline, neighbourhood models, and a latent factor model across a range of ranking metrics when recommending to groups as well as individual users. In ad-hoc retrieval, the IMM performs on par with the state-of-the-art IR ranking functions. Finally, we have demonstrated that we can build a preference based search engine using the IMF.

Chapter 6

Conclusion & Future Work

We started this thesis by describing the importance of relevance in information retrieval. We have also stated that relevance matching between two objects is achieved by matching the information on the objects. The most common and most successful way of estimating the relevance matching is by probabilistically modelling the objects and the relevance. The fundamental problem in probabilistic modelling lies in utilising the various kinds of information available in estimating the probability of relevance. A number of approaches, both probabilistic and non-probabilistic, have been proposed to solve the problem of information matching. However, the majority of these approaches compute the relevance matching (or information matching) when there is a specific kind of information available. For example, the ad-hoc text retrieval models are designed to use the common vocabulary between document-query pairs, and collaborative filtering techniques are used when there are no common features between user-item pairs. We discussed the most prominent of these approaches to solve the problem of information matching. The techniques used in these approaches fall into one of the following three categories: (a) estimating the relevance between two objects by representing them in a common *observable* feature space such as a vocabulary space, e.g. BM25, Language Models; (b) estimating the relevance by representing the objects in a common *hidden* feature space, e.g., the matrix factorisation methods used in recommendation; and (c) estimating the relevance by looking at the nearest neighbours of the object by computing explicit similarity, e.g. user-based and item-based collaborative filtering techniques. We have also presented how these techniques are not naturally designed to incorporate the information from different sources into the model while estimating the relevance. For example, in the matrix factorisation techniques used in solving a recommendation matching problem, any new information observed on the user or the item has to be represented in a hidden common feature. Similarly, in traditional probabilistic relevance ranking functions it is not straightforward to incorporate document or query specific information into the relevance estimation function. This problem of designing a formal probabilistic relevance model that can utilise the information on individual objects, their interaction and information from other similar objects, while estimating the relevance, was proposed by Robertson *et al.* in 1982 [RMC82a] and referred to as the unified model for information retrieval. The work in this thesis was primarily inspired by the problem of the unified model. And the goal of this thesis was to provide a fourth alternative, approach that could provide a solution to the problem.

In this thesis, we introduced a new theory called the *theory of information matching* (TIM) and a framework based on this theory called the *information matching framework* (IMF) which treats two distinct types of information objects as the objects generated from two distinct sets of feature spaces. This property allows the models on theory to represent the objects with their corresponding features' sets without making a restriction of representing them in a common feature set or space (observable or hidden). Under the TIM, we introduced the principle of *bi-directional matching* for relevance matching between two objects where they are relevant if and only if their bi-directional preferences are satisfied. In order to verify whether both of the objects satisfy the preferences of each other, firstly we defined the *bi-directional mappings* between the two feature sets where, these mappings are independent of the objects. Secondly, we used the *bi-directional mappings* between the features representing the objects to verify whether the *bi-directional preferences* are satisfied. This second property of separation between the object representations and preference mappings, along with the definition of relevance as a bi-directional preference matching, allows the models to modify the mappings and both object representations simultaneously. This property of the model is the essential component that was required to solve the problem of the unified model for information retrieval.

We demonstrated, theoretically, through some examples, and with some experiments how the IMF can be applied for the task of information matching in three different scenarios for the available information described with: (1) the available information on the interaction between the objects' and objects information is in two distinct feature spaces (e.g. collaborative filtering and group recommendation), (2) where the information available is in a common feature space with no interaction between the objects, e.g. ad-hoc retrieval with no relevance information, and (3) where the information available is from multiple sources (e.g. preference search engine). It is important to note that our primary objective for the thesis was to provide a theoretical framework for a formal probabilistic relevance model that can utilise multiple sources of information in computing the probability of relevance. To that end we provided new a new theory and framework and a Bayesian inference method for the probabilistic model developed in the framework. We believe that this thesis is a step towards the development of new theories to tackle information matching problems and to extract insights. We also believe that the amount of freedom that our theory provides us in modelling would open up a great number of applications and further theoretical developments in traditional information retrieval problems and beyond.

6.1 Limitations of the current work and future direction

The major limitation of the model developed on our framework in the current form is the high computation cost associated with the parameter estimation. As we described in section 5.4, the numbers of features and mappings grow with the data size unless there are some practical approximations to be made in reducing the feature space. We have built our first experimental preference search engine in section 5.4 by creating feature spaces based on observed data. This is a limitation of the implementation. However, if the complete model is utilised then it provides great a number of insights on objects as it learns information on all the possible distinct feature representations. For example, deploying the model on an advertising recommendation platform would provide a way to extract information on specific aspects of ads such as the probabilistic order of the features that represent the set of users who clicked on the ad. This sets the first important future direction as being to find a computationally efficient method for the information matching framework. Secondly, one could study the relationship between the information matching framework and the wider range of established probabilistic relevance ranking models. And the final important direction is to evaluate the theory and model in all possible scenarios of available information and to possibly find a way to integrate the concepts from this model into successful learning to rank method such as the LambdaRank [BRL06].

Appendix A

Optimisation in Bi-directional unified model

We can rewrite our optimisation objective function $\mathcal{F}\Big[q\big(\mathbf{\Theta}\big)\Big] = log(q(\mathbf{\Theta}))$ as follows:

$$
\begin{aligned}
\mathcal{F}\Big[q\big(\mathbf{\Theta}\big)\Big] \overset{def}{=} & \sum_m \sum_n \Bigg[r_{mn} log \prod_{k,l} z_{mn}^{kl} + (1-r_{mn}) log\Big(1 - \prod_{k,l} z_{mn}^{kl}\Big) \\
& + z_{mn}^{kl}\Big[\omega_k^m(1-\omega_l^n) log\ \big(1-\gamma_k^l\big) + \omega_l^n(1-\omega_k^m) log\ \big(1-\delta_l^k\big)\Big] \\
& + (1-z_{mn}^{kl})\Big[(1-\omega_k^m)\Big(log\beta_l^n + log\delta_l^k\Big) + \omega_k^m\big(log(1-\beta_l^n) + log\gamma_k^l\big)\Big]\Bigg] \\
& + \sum_m \Big[\omega_k^m\big[-\lambda_0^k + x_k^m log(\lambda_0^k) - log(x_k^m!)\big] + (1-\omega_k^m)\big[-\lambda_1^k + x_k^m log(\lambda_1^k) - log(x_k^m!)\big]\Big] \\
& + \sum_n \Big[\omega_l^n\big[-\lambda_0^l + y_l^n log(\lambda_0^l) - log(y_l^n!)\big] + (1-\omega_l^n)\big[-\lambda_1^l + y_l^n log(\lambda_1^l) - log(y_l^n!)\big]\Big] \\
& + \sum_m \sum_k \Big[\omega_k^m log(\alpha_k^m) + (1-\omega_k^m) log(1-\alpha_k^m)\Big] + \sum_n \sum_l \Big[\omega_n^l log(\beta_l^n) + (1-\omega_n^l) log(1-\beta_l^n)\Big] \\
& + \sum_m \sum_k \theta^1_{\alpha_k^m} log(\alpha_k^m) + \theta^2_{\alpha_k^m} log(1-\alpha_k^m) + \sum_n \sum_l \theta^1_{\beta_l^n} log(\beta_l^n) + \theta^2_{\beta_l^n} log(1-\beta_l^n) \\
& + \sum_k \sum_l \Big[\theta^1_{\gamma_k^l} log(\gamma_k^l) + \theta^2_{\gamma_k^l} log(1-\gamma_k^l) + \theta^1_{\delta_l^k} log(\delta_l^k) + \theta^2_{\delta_l^k} log(1-\delta_l^k)\Big] \\
& + \sum_k \Bigg(\big[a_k^0 log(b_k^0) + (a_k^0-1) log\lambda_k^0 - b_k^0\lambda_k^0 - log\,\Gamma(a_k^0)\big] \\
& + \big[a_k^1 log(b_k^1) + (a_k^1-1) log\lambda_k^1 - b_k^1\lambda_k^1 - log\,\Gamma(a_k^1)\big]\Bigg) \\
& + \sum_l \Bigg(\big[a_l^0 log(b_l^0) + (a_l^0-1) log\lambda_l^0 - b_l^0\lambda_l^0 - log\,\Gamma(a_l^0)\big] \\
& + \big[a_l^1 log(b_l^1) + (a_l^1-1) log\lambda_l^1 - b_k^1\lambda_l^1 - log\,\Gamma(a_l^1)\big]\Bigg) \\
& + \sum_m \sum_n \theta_m^n(1) log(\prod_{ij} z_{mn}^{kl}) + \theta_m^n(2) log(1 - \prod_{ij} z_{mn}^{kl})
\end{aligned}
$$

A.0.1 Deriving parameter update equations

We estimate ω_k^m, ω_l^n, α_k^m, β_l^n, γ_k^l, δ_l^k as well as λ_0^k, λ_1^k, λ_0^l, λ_1^l and z_{mn}^{kl} by computing the functional derivatives $\dfrac{\partial\mathcal{F}}{\partial q}$ with respect to each distribution in $q(\mathbf{\Theta})$ and setting it equals to zero. By setting the derivatives to zero we get the update steps for each distribution in $q(\mathbf{\Theta})$ in terms of their sufficient statistics. In what follows, we derive update steps for each approximated factor distribution.

Updating $q(\alpha_k^m)$

By equating $\frac{\partial \mathcal{F}}{\partial q(\alpha_k^m)} = 0$, we get

$$\left[\langle \omega_k^m \rangle_q + \theta^1_{\alpha_k^m}\right] \langle log(\alpha_k^m) \rangle_q + \left[\theta^2_{\alpha_k^m} + (1 - \langle \omega_k^m \rangle_q)\right] \langle log(1-\alpha_k^m) \rangle_q \; + \; const = 0 \tag{A.1}$$

The result is a beta distribution with natural parameters defined in terms of their sufficient statistics where,

$$\theta^1_{\alpha_k^m}(new) = \langle \omega_k^m \rangle_q + \theta^1_{\alpha_k^m}$$
$$\theta^2_{\alpha_k^m}(new) = (1 - \langle \omega_k^m \rangle_q) + \theta^2_{\alpha_k^m}$$

Now, the expected value of $log\ \alpha_k^m$ and $log(1-\alpha_k^m)$ under q distribution can be computed as

$$\langle log\ \alpha_k^m \rangle_q = \Psi\big(\theta^1_{\alpha_k^m}(new)\big) - \Psi\big(\theta^1_{\alpha_k^m}(new) + \theta^2_{\alpha_k^m}(new)\big)$$
$$\langle log(1-\alpha_k^m) \rangle_q = \Psi\big(\theta^2_{\alpha_k^m}(new)\big) - \Psi\big(\theta^1_{\alpha_k^m}(new) + \theta^2_{\alpha_k^m}(new)\big)$$

where Ψ is a digamma function and defined as

$$\Psi(c) = \frac{d}{dc}\ ln(\Gamma(c)) \tag{A.2}$$

Updating $q(\beta_l^n)$

By equating $\frac{\partial \mathcal{F}}{\partial q(\beta_l^n)} = 0$, we get

$$\left[\langle \omega_l^n \rangle_q + \theta^1_{\beta_l^n} + \sum_m \sum_k (1 - \langle z_{mn}^{kl} \rangle_q)(1 - \langle \omega_k^m \rangle_q)\right] \langle log(\beta_l^n) \rangle_q$$
$$+ \left[\theta^2_{\beta_l^n} + (1 - \langle \omega_l^n \rangle_q) + \sum_m \sum_k (1 - \langle z_{mn}^{kl} \rangle_q) \langle \omega_k^m \rangle_q\right] \langle log(1-\beta_l^n) \rangle_q + const = 0$$

Similarly, the result is a beta distribution with the natural parameters are defined in terms of their sufficient statistics where,

$$\theta^1_{\beta_l^n}(new) = \langle \omega_l^n \rangle_q + \theta^1_{\beta_l^n} + \sum_m \sum_k (1 - \langle z_{mn}^{kl} \rangle_q)(1 - \langle \omega_k^m \rangle_q)$$
$$\theta^2_{\beta_l^n}(new) = \theta^2_{\beta_l^n} + (1 - \langle \omega_l^n \rangle_q) + \sum_m \sum_k (1 - \langle z_{mn}^{kl} \rangle_q) \langle \omega_k^m \rangle_q$$

The expected value of $log\ \beta_l^n$ and $log(1-\beta_l^n)$ under q distribution can be computed as

$$\langle log \beta_l^n \rangle_q = \Psi\big(\theta^1_{\beta_l^n}(new)\big) - \Psi\big(\theta^1_{\beta_l^n}(new) + \theta^2_{\beta_l^n}(new)\big)$$
$$\langle log(1-\beta_l^n) \rangle_q = \Psi\big(\theta^2_{\beta_l^n}(new)\big) - \Psi\big(\theta^1_{\beta_l^n}(new) + \theta^2_{\beta_l^n}(new)\big)$$

Updating $q(\gamma_k^l)$ and $q(\delta_l^k)$

Similar to α_k^m and β_l^n, by setting $\frac{\partial \mathcal{F}}{\partial q(\gamma_k^l)} = 0$ and $\frac{\partial \mathcal{F}}{\partial q(\delta_l^n)} = 0$ we get beta distributions as follows:

$$log(\gamma_k^l)\left[\sum_m \sum_n (1 - \langle z_{mn}^{kl} \rangle_q) \langle \omega_k^m \rangle_q + \theta^1_{\gamma_k^l}\right] + log(1-\gamma_k^l)\left[\sum_m \sum_n \langle z_{mn}^{kl} \rangle_q \left[\langle \omega_k^m \rangle_q (1 - \langle \omega_l^n \rangle_q)\right] + \theta^2_{\gamma_k^l}\right] = 0 \tag{A.3}$$

$$\begin{aligned}
\theta^1_{\gamma^l_k}(new) &= \sum_m \sum_n (1 - \left\langle z^{kl}_{mn} \right\rangle_q) \left\langle \omega^m_k \right\rangle_q + \theta^1_{\gamma^l_k} \\
\theta^2_{\gamma^l_k}(new) &= \sum_m \sum_n \left\langle z^{kl}_{mn} \right\rangle_q \left[\left\langle \omega^m_k \right\rangle_q (1 - \left\langle \omega^n_l \right\rangle_q) \right] + \theta^2_{\gamma^l_k} \\
\left\langle log(\gamma^l_k) \right\rangle_q &= \Psi(\theta^1_{\gamma^l_k}(new)) - \Psi\big(\theta^1_{\gamma^l_k}(new) + \theta^2_{\gamma^l_k}(new)\big) \\
\left\langle log(1 - \gamma^l_k) \right\rangle_q &= \Psi(\theta^2_{\gamma^l_k}(new)) - \Psi\big(\theta^1_{\gamma^l_k}(new) + \theta^2_{\gamma^l_k}(new)\big)
\end{aligned}$$

$$log(\delta^k_l)\left[\sum_m \sum_n (1 - \left\langle z^{kl}_{mn} \right\rangle_q)(1 - \left\langle \omega^m_k \right\rangle_q) + \theta^1_{\delta^k_l}\right] + log(1 - \delta^k_l)\left[\sum_m \sum_n \left\langle z^{kl}_{mn} \right\rangle_q \left[(1 - \left\langle \omega^m_k \right\rangle_q) \left\langle \omega^n_l \right\rangle_q\right] + \theta^2_{\delta^k_l}\right] = 0 \tag{A.4}$$

$$\begin{aligned}
\theta^1_{\delta^k_l}(new) &= \sum_m \sum_n (1 - \left\langle z^{kl}_{mn} \right\rangle_q)(1 - \left\langle \omega^m_k \right\rangle_q) + \theta^1_{\delta^k_l} \\
\theta^2_{\delta^k_l}(new) &= \sum_m \sum_n \left\langle z^{kl}_{mn} \right\rangle_q \left[(1 - \left\langle \omega^m_k \right\rangle_q) \left\langle \omega^n_l \right\rangle_q\right] + \theta^2_{\delta^k_l} \\
\left\langle log(\delta^k_l) \right\rangle_q &= \Psi(\theta^1_{\delta^k_l}(new)) - \Psi\big(\theta^1_{\gamma^l_k}(new) + \theta^1_{\delta^k_l}(new)\big) \\
\left\langle log(1 - \delta^k_l) \right\rangle_q &= \Psi(\theta^2_{\delta^k_l}(new)) - \Psi\big(\theta^1_{\delta^k_l}(new) + \theta^2_{\delta^k_l})(new)\big)
\end{aligned}$$

Updating $q(\omega^m_k = 1)$ and $q(\omega^n_l = 1)$

By equating $\dfrac{\partial \mathcal{F}}{\partial q(\omega^m_k)} = 0$ we get

$$\begin{aligned}
&\omega^m_k \Bigg(\sum_n \sum_l \left\langle z^{kl}_{mn} \right\rangle_q \Big[(1 - \left\langle \omega^n_l \right\rangle_q) \left\langle log(1 - \gamma^l_k) \right\rangle_q + \left\langle log(\delta^k_l) \right\rangle_q \big] + \left\langle log(\beta^n_l) \right\rangle_q \\
&- \left\langle log(\gamma^l_k) \right\rangle_q - \left\langle \omega^n_l \right\rangle_q \left\langle log(1 - \delta^k_l) \right\rangle_q - \left\langle log(1 - \beta^n_l) \right\rangle_q \Big] \\
&+ \big(\left\langle \lambda^k_1 \right\rangle_q - \left\langle \lambda^k_0 \right\rangle_q \big) + x^m_k \big(\left\langle log(\lambda^k_0) \right\rangle_q - \left\langle log(\lambda^k_1) \right\rangle_q \big) + \left\langle log(\alpha^m_k) \right\rangle_q - \left\langle log(1 - \alpha^m_k \right\rangle_q) \Bigg) + const = 0
\end{aligned}$$

We compute $q(\omega^m_k = 1)$ as

$$q(\omega^m_k = 1) = \frac{1}{(1 + e^{-w^m_k(0)})} \tag{A.5}$$

where

$$\begin{aligned}
w^m_k(0) = \Bigg(&\sum_n \sum_l \left\langle z^{kl}_{mn} \right\rangle_q \Big[(1 - \left\langle \omega^n_l \right\rangle_q) \left\langle log(1 - \gamma^l_k) \right\rangle_q + \left\langle log(\delta^k_l) \right\rangle_q \big] + \left\langle log(\beta^n_l) \right\rangle_q \\
&- \left\langle log(\gamma^l_k) \right\rangle_q - \left\langle \omega^n_l \right\rangle_q \left\langle log(1 - \delta^k_l) \right\rangle_q - \left\langle log(1 - \beta^n_l) \right\rangle_q \Big] \\
&+ \big(\left\langle \lambda^k_1 \right\rangle_q - \left\langle \lambda^k_0 \right\rangle_q \big) + x^m_k \big(\left\langle log(\lambda^k_0) \right\rangle_q - \left\langle log(\lambda^k_1) \right\rangle_q \big) + \left\langle log(\alpha^m_k) \right\rangle_q - \left\langle log(1 - \alpha^m_k \right\rangle_q) \Bigg)
\end{aligned}$$

Similarly, we compute $q(\omega_l^n = 1)$ by setting $\frac{\partial \mathcal{F}}{\partial q(\omega_l^n)} = 0$ and using

$$q(\omega_l^n = 1) = \frac{1}{(1 + e^{-w_l^n(0)})} \tag{A.6}$$

where

$$\omega_l^n(0) = \sum_m \sum_k \langle z_{mn}^{kl} \rangle_q \left[\langle log(1-\delta_l^k) \rangle_q - \langle \omega_k^m \rangle_q \left(\langle log(1-\gamma_k^l) \rangle_q + \langle log(1-\delta_l^k) \rangle_q \right) \right]$$
$$+ \left(\lambda_1^n - \lambda_0^n\right) + y_l^n \left(\langle log(\lambda_0^l) \rangle_q - \langle log(\lambda_1^l) \rangle_q \right) + \langle log(\beta_l^n) \rangle_q - \langle log(1-\beta_l^n) \rangle_q$$

Updating $q(\lambda_0^k)$, $q(\lambda_1^k)$, $q(\lambda_0^l)$ and $q(\lambda_1^l)$

By equating $\frac{\partial \mathcal{F}}{\partial q(\lambda_0^k)} = 0$ we get,

$$log(\lambda_0^k) \left[a_0^k - 1 + \sum_m \langle \omega_k^m \rangle_q x_k^m \right] - \lambda_0^k \left[b_0^k + \sum_m \langle \omega_k^m \rangle_q \right] = 0 \tag{A.7}$$

The result is a gamma distribution with their natural parameters described in terms of sufficient statistics. The parameters of the distribution are given by,

$$a_0^k(new) = a_0^k - 1 + \sum_m \langle \omega_k^m \rangle_q x_k^m$$
$$b_0^k(new) = b_0^k + \sum_m \langle \omega_k^m \rangle_q$$

The expected value of $log\ \lambda_0^k$ under $q(\lambda_0^k)$ is give by

$$\langle log\ \lambda_0^k \rangle_q = \Psi(a_0^k(new)) - log(b_0^k(new)) \tag{A.8}$$

Similarly, by setting $\frac{\partial \mathcal{F}}{\partial q(\lambda_1^k)} = 0$, $\frac{\partial \mathcal{F}}{\partial q(\lambda_0^l)} = 0$ and $\frac{\partial \mathcal{F}}{\partial q(\lambda_1^l)} = 0$ we get

$$log(\lambda_1^k) \left[a_1^m - 1 + \sum_m x_k^m (1 - \langle \omega_k^m \rangle_q) \right] - \lambda_1^k \left[b_1^k + \sum_m (1 - \langle \omega_k^m \rangle_q) \right]$$
$$log(\lambda_0^l) \left[a_0^l - 1 + \sum_m \langle \omega_l^n \rangle_q y_l^n \right] - \lambda_0^l \left[b_0^l + \sum_m \langle \omega_l^n \rangle_q \right]$$
$$log(\lambda_1^l) \left[a_1^l - 1 + y_l^n \sum_m (1 - \langle \omega_l^n \rangle_q) \right] + \lambda_1^l \left[b_1^l + \sum_n (1 - \langle \omega_l^n \rangle_q) \right]$$

And,

$$a_1^k(new) = \left[a_1^k - 1 + \sum_m x_k^m (1 - \langle \omega_k^m \rangle_q) \right] \text{ and } b_1^k(new) = b_1^k + \sum_m (1 - \langle \omega_k^m \rangle_q)$$
$$a_0^l(new) = \left[a_0^l - 1 + \sum_n \langle \omega_l^n \rangle_q y_l^n \right] \text{ and } b_0^l(new) = b_0^l + \sum_n \langle \omega_l^n \rangle_q$$
$$a_1^l(new) = \left[a_1^l - 1 + y_l^n \sum_n (1 - \langle \omega_l^n \rangle_q) \right] \text{ and } b_1^l(new) = \mathbf{b}_1^l + \sum_n (1 - \langle \omega_l^n \rangle_q)$$

And,

$$\langle log\lambda_0^k \rangle_q = \Psi(a_1^m(new)) - log(b_1^k(new))$$
$$\langle log\lambda_0^l \rangle_q = \Psi(a_0^l(new)) - log(b_0^l(new))$$
$$\langle log\lambda_1^l \rangle_q = \Psi(a_1^l(new)) - log(a_1^l(new))$$

Updating $q(\mathbf{z}_{mn})$ and $q(z_{mn}^{kl} = 1)$

By equating $\dfrac{\partial \mathcal{F}}{\partial q(\mathbf{z}_{mn})} = 0$, we get

$$\Big(\Theta_m^n(1) + r_{mn}\Big) log(\prod_{kl} z_{mn}^{kl}) + \Big(\Theta_m^n(2) + 1 - r_{mn}\Big) log(1 - \prod_{kl} z_{mn}^{kl}) = 0$$

Again, the result is a beta distribution with the natural parameters described in terns of sufficient statistics. The new parameters of the distribution are

$$\Theta_m^n(1)(new) = \Theta_m^n(1) + r_{mn}$$
$$\Theta_m^n(2)(new) = \Theta_m^n(2) + (1 - r_{mn})$$

And the expected values of $log\ \big(\mathbf{z}_{mn}\big)$

$$\langle log\ \mathbf{z}_{mn} \rangle_q = \Psi(\Theta_m^n(1)(new)) - \Psi(\Theta_m^n(1)(new) + \Theta_m^n(2)(new))$$

We can write $q(z_{mn}^{kl} = 1) = w_z + const$. That is,

$$\begin{aligned} z_{mn}^{kl} \Bigg[& \langle \omega_k^m \rangle_q \Big((1 - \langle \omega_l^n \rangle_q) \left\langle log(1 - \gamma_k^l) \right\rangle_q - \left\langle log(\gamma_k^l) \right\rangle_q - \langle log(1 - \beta_l^n) \rangle_q \Big) \\ & + (1 - \langle \omega_k^m \rangle_q) \Big[\langle \omega_l^n \rangle_q \langle log(1 - \delta_l^n) \rangle_q - \langle log(\beta_l^n) \rangle_q - \left\langle log(\delta_l^k) \right\rangle_q \Big] \Bigg] \\ & + const = 0 \end{aligned}$$

We compute the new z_{mn}^{kl} as follows:

$$q(z_{mn}^{kl} = 1) = \frac{1}{(1 + e^{-w_z})} \tag{A.9}$$

where

$$\begin{aligned} w_z = \Bigg[& \langle \omega_k^m \rangle_q \Big((1 - \langle \omega_l^n \rangle_q) \left\langle log(1 - \gamma_k^l) \right\rangle_q - \left\langle log(\gamma_k^l) \right\rangle_q - \langle log(1 - \beta_l^n) \rangle_q \Big) \\ & + (1 - \langle \omega_k^m \rangle_q) \Big(\langle \omega_l^n \rangle_q \langle log(1 - \delta_l^n) \rangle_q - \langle log(\beta_l^n) \rangle_q - \left\langle log(\delta_l^k) \right\rangle_q \Big) \Bigg] \end{aligned}$$

Bibliography

[AD11] Robin Aly and Thomas Demeester. Towards a better understanding of the relationship between probabilistic models in ir. In *Proceedings of the Third international conference on Advances in information retrieval theory*, ICTIR'11, pages 164–175, Berlin, Heidelberg, 2011. Springer-Verlag.

[AYRC$^+$09] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, August 2009.

[BCC11] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *RecSys*, 2011.

[BF10] Shlomo Berkovsky and Jill Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 111–118, New York, NY, USA, 2010. ACM.

[BHC98] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[BMR10] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 119–126, New York, NY, USA, 2010. ACM.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3, March 2003.

[BR04] David Bodoff and Stephen Robertson. A new unified probabilistic model. *J. Am. Soc. Inf. Sci. Technol.*, 55(6):471–487, April 2004.

[BRL06] Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In *NIPS*, pages 193–200, 2006.

[BV09] Corrado Boscarino and Arjen P. Vries. Prior information and the determination of event spaces in probabilistic information retrieval models. In *Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory*, ICTIR '09, pages 257–264, Berlin, Heidelberg, 2009. Springer-Verlag.

[CBH02] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 184–185, New York, NY, USA, 2002. ACM.

[CK68] C. Cleverdon and M. Kean. Factors determining the performance of indexing systems. Aslib Cranfield Research Project, Cranfield, England, 1968.

[CKT10] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, RecSys '10, 2010.

[CZL$^+$11] Tianqi Chen, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. Feature-based matrix factorization. *CoRR*, abs/1109.2271, 2011.

[DK04] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, January 2004.

[DKNS01] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society, Series B*, 1977.

[DR94] Jean Diebolt and Christian P. Robert. Estimation of finite mixture distributions through bayesian sampling. *J. R. Statist. Soc B*, (2):363–375, 1994.

[FB91] Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Trans. Inf. Syst.*, 9(3):223–248, July 1991.

[Fuh86] Norbert Fuhr. Two models of retrieval with probabilistic indexing. In *SIGIR'86, Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, September 8-10, 1986*, pages 249–257. ACM, 1986.

[Fuh92] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35:243–255, 1992.

[GHB13] Prem Gopalan, Jake M. Hofman, and David M. Blei. Scalable recommendation with poisson factorization. *CoRR*, abs/1311.1704, 2013.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70, December 1992.

[Har75a] S. P. Harter. A probabilistic approach to automatic keyword indexing. *Journal of the American Society for Information Retrieval Science*, 1975.

[Har75b] S. P. Harter. A probabilistic approach to automatic keyword indexing. part i: On the distribution of specialty words in a technical literature. *Journal of the American Society for Informaiton Science*, 1975.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, January 2004.

[Hof99] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, SIGIR '99, 1999.

[Hof04] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 2004.

[Jon72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. 1972.

[JS07] Anthony Jameson and Barry Smyth. The adaptive web. chapter Recommendation to groups, pages 596–627. Springer-Verlag, Berlin, Heidelberg, 2007.

[KBV09] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 2009.

[Lav04] Victor Lavrenko. *A Generative Theory of Relevance*. Ir, University of Massachusetts, 2004.

[Luk08] Robert W. Luk. On event space and rank equivalence between probabilistic retrieval models. *Inf. Retr.*, 11(6):539–561, December 2008.

[LX14] Hang Li and Jun Xu. Semantic matching in search. *Foundations and Trends in Information Retrieval*, 7(5):343–469, 2014.

[LZ01] John Lafferty and ChengXiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, pages 111–119, 2001.

[LZ03] John Lafferty and ChengXiang Zhai. Probabilistic relevance models based on document and query generation. In W.Bruce Croft and John Lafferty, editors, *Language Modeling for Information Retrieval*, volume 13 of *The Springer International Series on Information Retrieval*, pages 1–10. Springer Netherlands, 2003.

[LZ11] Yuanhua Lv and ChengXiang Zhai. Lower-bounding term frequency normalization. In *CIKM*, CIKM '11, 2011.

[Mac02] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[MAL$^+$03] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 263–266, New York, NY, USA, 2003. ACM.

[Mas04] Judith Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85, February 2004.

[MFZ07] Qiaozhu Mei, Hui Fang, and ChengXiang Zhai. A study of poisson query generation model for information retrieval. In *In SIGIR*, SIGIR '07, 2007.

[Miz97] S Mizarro. Relevance: The whole history. *Journal of the American Society for Information Science*, 48(9):321–343, 1997.

[MK60a] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 7(3):216–244, July 1960.

[MK60b] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *J. ACM*, 1960.

[MSC$^+$06] Kevin McCarthy, Maria Salamó, Lorcan Coyle, Lorraine McGinty, Barry Smyth, and Paddy Nixon. Cats: A synchronous approach to collaborative group recommendation. In *FLAIRS Conference*, pages 86–91, 2006.

[OCKR01] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: a recommender system for groups of users. In *Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work*, ECSCW'01, pages 199–218, Norwell, MA, USA, 2001. Kluwer Academic Publishers.

[PB07] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *SIGIR*, 1998.

[PDCCA05] Sebastiano Pizzutilo, Berardina De Carolis, Giovanni Cozzolongo, and Francesco Ambruoso. Group modeling in a public space: methods, techniques, experiences. In *Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, AIC'05, pages 175–180, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS).

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[RMC82a] SE Robertson, M. E. Maron, and W. S. Cooper. Probability of relevance: a unification of two competing models for document retrieval. *Information Technology: Research and Development*, 1982.

[RMC82b] Stephen E. Robertson, M. E. Maron, and William S. Cooper. The unified probabilistic model for ir. In *SIGIR*, pages 108–117, 1982.

[Rob97a] S. E. Robertson. Overview of the Okapi projects. *Journal of Documentation*, 53:3–7, 1997.

[Rob97b] S. E. Robertson. The probability ranking principle in ir. pages 281–286, 1997.

[Rob03] Stephen Robertson. The unified model revisited. In *SIGIR 2003 Workshop on Mathematical/Formal Models in Information Retrieval*, 2003.

[Rob04] Stephen Robertson. On event spaces and probabilistic models in information retrieval. In *SIGIR 2002 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2004.

[Rob05] Stephen Robertson. On event spaces and probabilistic models in information retrieval. *Inf. Retr.*, 8(2):319–329, April 2005.

[Roc71] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[Röl13] Thomas Rölleke. *Information Retrieval Models: Foundations & Relationships*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2013.

[RSJ76] S.E Robertson and K Spark Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 1976.

[RTK06] Thomas Rölleke, Theodora Tsikrika, and Gabriella Kazai. A general matrix framework for modelling information retrieval. *Inf. Process. Manage.*, 42(1):4–30, 2006.

[RvRP80] Stephen E. Robertson, C. J. van Rijsbergen, and Martin F. Porter. Probabilistic models of indexing and searching. In *SIGIR*, 1980.

[RW94] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, 1994.

[RZ09] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 2009.

[Sai11] *CAMRa '11: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, New York, NY, USA, 2011. ACM.

[SCTB$^+$12] David Sontag, Kevyn Collins-Thompson, Paul N. Bennett, Ryen W. White, Susan Dumais, and Bodo Billerbeck. Probabilistic models for personalizing web search. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 433–442, New York, NY, USA, 2012. ACM.

[Sin97] Amitabh Kumar Singhal. *Term weighting revisited.* PhD thesis, Ithaca, NY, USA, 1997. UMI Order No. GAX97-14899.

[SK09] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, pages 4:2–4:2, January 2009.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[SM95] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating ẁord of mouth,́ In *CHI'95*, 1995.

[TC90] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '90, pages 1–24, New York, NY, USA, 1990. ACM.

[vR86] C. J. van Rijsbergen. A non-classical logic for information retrieval. *Comput. J.*, 29(6):481–485, 1986.

[WdVR06] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM.

[WdVR08] Jun Wang, Arjen P. de Vries, and Marcel J.T. Reinders. Unified relevance models for rating prediction in collaborative filtering. *ACM Trans. on Information System (TOIS)*, 2008.

[WMR96] Steve Waterhouse, David Mackay, and Tony Robinson. Bayesian methods for mixtures of experts. In *In*, pages 351–357. MIT Press, 1996.

[WY91] S. K. M. Wong and Y. Y. Yao. A probabilistic inference model for information retrieval. *Inf. Syst.*, 16(3):301–321, July 1991.

[WZ10] Jun Wang and Jianhan Zhu. On statistical analysis and optimization of information retrieval effectiveness metrics. In *SIGIR*, 2010.

[YZHG06] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. TV program recommendation for multiple viewers based on user profile merging. *User Modeling and User-Adapted Interaction*, 16(1):63–82, March 2006.

[Zha08] ChengXiang Zhai. Statistical language models for information retrieval: Critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.

[ZL04] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.