FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Aydin Ahmadli

Probabilistic Models for Recommender Systems

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Mgr. Marta Vomlelová, Ph.D.

Study programme: Computer Science

Specialization: Artificial Intelligence

Prague 2022

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In…....... date............   signature

Title: Probabilistic Models for Recommender Systems

Author: Aydin Ahmadli

Department / Institute: Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Mgr. Marta Vomlelová Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Recommender systems are software tools and techniques providing recommendations to users based on their needs. Today, popular e-commerce sites widely use recommender systems to recommend product items, articles, books, music, etc. In this thesis, we discuss various probabilistic models for recommender systems, and put the most focus on implementation of hybrid and interpretable probabilistic content-based collaborative filtering model, called Collaborative Topic model for Poisson distributed ratings (CTMP) augmented with Bernoulli randomness for Online Maximum a Posteriori Estimation (BOPE). Resulting model outperforms the previously existing models significantly with its main competency being in commercial product recommendations. It is a fast, scalable, and efficient in ill-posed cases, including short text and sparse data. The model is trained and tested on well-known MovieLens 20M and NETFLIX datasets, and empirical evaluations such as recall, precision, sparsity and topic interpretations are promising.

Keywords: probabilistic-models, recommender-systems, topic-modeling, variational-inference

Contents

1 Introduction

Since the mid-1990s, when the first articles on collaborative filtering were published, Recommender Systems (RS) have been an active study topic. In recent years, many websites utilize the RS extensively. "You may like this", "Customers who purchased this item also purchased", and "Other products you may like". Everyone has at least once encountered these recommendations when exploring the web pages on internet. These are the capabilities of RS.

In this thesis, we will explore a variety of probabilistic models developed for RS. Our study focuses mostly on implementation of Collaborative Topic model for Poisson distributed ratings (CTMP) [1] augmented with Bernoulli randomness for Online Maximum a Posteriori Estimation (BOPE) [28]. CTMP is a probabilistic hybrid model with scalability and interpretability. Its key properties make it advantageous compared to its predecessors. BOPE is the recently proposed Maximum a Posteriori Estimation (MAP) algorithm that provides a fast convergence rate along with implicit regularization. These features help probabilistic models to excel in ill-posed cases such as training on short text, and sparse or noisy data. In this thesis, we will be implementing and evaluating CTMP model that uses BOPE for its MAP algorithm.

In the following sections 2 and 3, we will introduce RS and probabilistic models for RS. Then we will continue with Latent Dirichlet Allocation (LDA) and Variational Inference (VI) in sections 4 and 5. Note that LDA and VI concepts are crucial parts of the CTMP model theory. Next, in section 0, we will discuss Online Maximum a Posteriori Estimation (OPE) and BOPE for solving MAP problem. Finally, sections 7 and 8 are dedicated to CTMP model and its empirical studies.

2 Recommender Systems

Recommender Systems (RS) are widely recognized as one of the most beneficial applications of Machine Learning. The fundamental objective of these Machine Learning-driven Recommenders is to filter, prioritize and efficiently deliver the necessary information to the consumers amid overwhelmingly numerous choices on the internet. It is also described as:

"Any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options." [2]

Therefore, many companies utilize RS to help consumers discover new and relevant items such as movies, songs, jobs, etc. They use the consumer data in the explicit or implicit form (e.g., likes, clicks), to comprehensively assess consumers' preferences and then recommend the relevant items to them. According to various criteria, there are multiple techniques of RS, each of which differs in how a single recommendation is generated. The most common types of RS are described in the following sections.

Although there exist several different RS in the literature, we will focus on the three most common techniques as shown in Figure *2.1*:
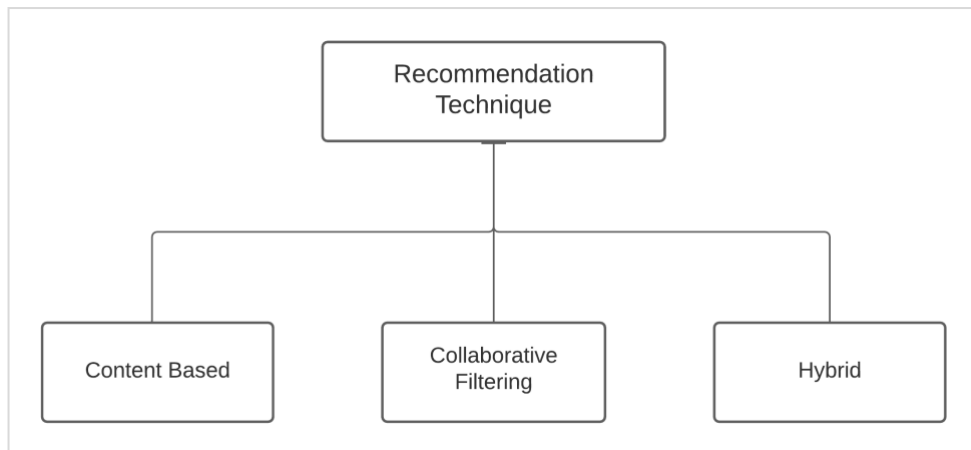
Figure 2.1: Types of recommendation technique.

2.1 Collaborative Filtering recommender systems

Collaborative Filtering (CF) recommender systems are one of the most widely used systems next to the Content Based (CB) recommender systems. Essentially, these systems create a user profile based on the ratings of various items and then aim to compare these against a wider user group [3]. As the word "collaborative" from the name implies, multiple users come together as a group – a taste of one user will be similar to the other users of the group. Therefore, by utilizing the user's data which contains their historical preferences on a set of items, the system deploys an assumption that the users who have previously agreed are more likely to agree again in the future. So, the system creates new recommendations by taking the similarities between users based on the ratings into consideration (see Figure *2.2*).
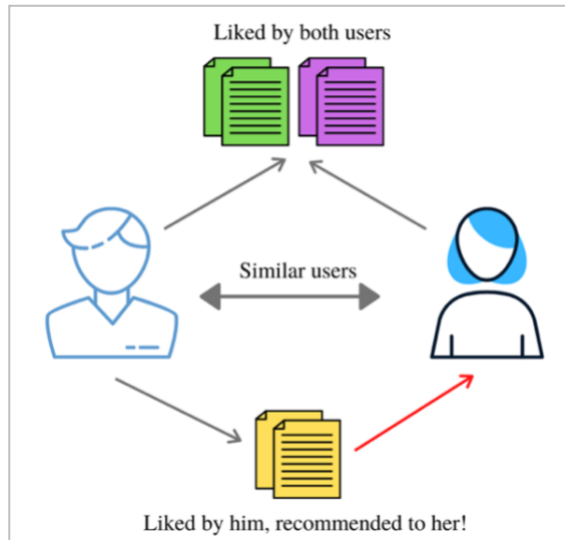
Figure 2.2: Illustration of Collaborative Filtering recommender systems.

Although CF recommender systems have been used in the industry for many years [4], they still have a limitation such that they cannot address the cold start problem – they are not able to recommend items which are not rated by any users (e.g., new items). As a result, only famous items may get recommended. Furthermore, traditional CF systems are also memory-wise and computationally expensive and suffer from scalability problems.

2.2 Content Based recommender systems

While CF recommender systems, as discussed above, recommend the products or items according to the similarities of user preferences which means that recommendation relies on the user-item interactions, CB recommender systems, on the other hand, aim to recommend products or items similar to those a given user has rated positive or liked in the past. So, CB systems generate recommendations based on the comparison between the content of the items and the user profile which was created according to the historical user data (see Figure *2.3*). Note that the content of items is described by terms, tags, features, or even plots in case the items are movies.

An algorithm used to recommend the movies on the Netflix platform is a prominent example that resembles these RS. If a certain user watches and comedy movie and rates it positive via votes or comments, then the new movie recommendations with the same label as that liked movie will be suggested to the user. In other words, based on the content of the consumed item, this RS finds other similar items and recommends them. Note that such website platforms often keep the techniques of how the content is labeled and matched against each other as private [3]. Contrary to CF systems, CB system doesn't suffer from a cold-start problem, and they can suggest not only famous or older items but also unpopular or new items. In addition to this, they are memory-wise and computationally cheap because there is no need for the data of other users to be able to compute the recommendation for a specific user.
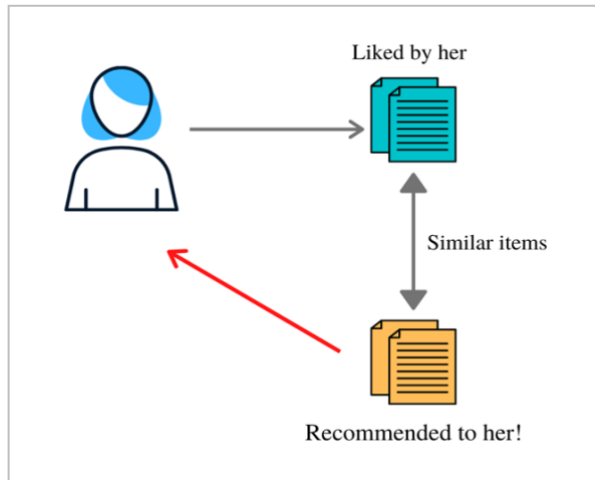
Figure 2.3: Illustration of Content Based recommender systems.

2.3 Hybrid recommender systems

Hybrid recommender systems combine two or more types of traditional RS to have better performance by benefiting from complementary advantages of subsystems. Hybrid systems which combine CF and CB approaches achieve state-of-the-art results in many cases and are used in many large-scale RS nowadays. Detailed comparison of advantages and disadvantages of Hybrid Recommenders along with CF and CB Recommenders are shown in Table *2.1* below**:**

	Collaborative Filtering	Content Based	Hybrid
Number of users	Recommendation based on many users having similar interest	Recommendation based on single user	Combination of collaborative and content based filtering
Disadvantages	- Cold start problem - Data sparsity - Scalability - Memory-wise and computationally expensive	- Limited content analysis - Over-specialization	- Increased complexity - Increased expense of implementation
Advantages	- Serendipitous recommendation - User and item features are not required - Quality may improve over time as more users interact with items - - Minimal domain knowledge required	- User independent - No cold start problem - Interpretable results - Memory-wise and computationally cheap	Avoids most of the shortcomings of other approaches.

Table 2.1: Detailed comparison of recommender systems.

3 Probabilistic Models for Recommender Systems

The application of probabilistic modeling to the recommendation problem has a rich history that dates back decades. Many authors incorporated the probabilistic approaches into models which explained the dataset. Initial approaches were probabilistic graphical models such as Bayesian networks and Dependency networks which eventually left their place with subsequent novel topic models such as Latent Dirichlet Allocation (LDA) [5]. The term "latent" is used in their name because LDA is considered a probabilistic topic model, and the topics it aims to find from the corpus are considered latent or hidden variables. A detailed explanation of LDA has been discussed in section 4 as it is an essential part of the model that we will be discussing in this thesis. Also, note that, as LDA can suggest items that have similar content to other items that a user likes, it has been extensively used for CB recommender systems. When it comes to the field of CF recommender systems, the matrix factorization technique had gained decent popularity, especially after being combined with a probabilistic approach [6], [7], [8].

3.1 fLDA

Lately, there has been a lot of interest in combining probabilistic topic modeling with matrix factorization in the field of hybrid recommender systems. One of the major reasons for this is that when the content of an item is represented by topic models, the models benefit from interpretable semantics of the latent space characterized by the topic mixtures, and this leads to more interpretable semantics of the item latent factor. Initially, Deepak Agarwal and Bee-Chung Chen proposed Matrix Factorization through Latent Dirichlet Allocation (fLDA) where the item latent factor took the role of topic proportion in the LDA representation [9]. Despite being an accurate and interpretable model, which handles both cold-start and warm-start scenarios, fLDA still had a limitation in dealing with distinguished items where there is an identical topic mixture, but content details that topic mixture cannot cover are of concern to different groups of people. To elaborate on this limitation, consider that we have two articles: A and B. Both articles are about the application of machine learning to social networks. Because both articles are identical in terms of their contents, they will also possess the same topic proportions. Now let's consider that these two articles are of interest to different kinds of users: Article A provides a prominent machine learning algorithm that is applied to social network applications, whereas article B implements a standard machine learning algorithm, but provides crucial data analysis on social network data. As a result, users who work in machine learning will prefer article A and will hardly be interested in article B, whereas users who work in social networks will be more interested in article B instead of A. However, as the topic proportions of both articles are the same, both will be recommended to both groups of users [10].

3.2 CTR

To tackle the limitation mentioned above, a novel approach called Collaborative Topic Regression (CTR) has been proposed by David M. Blei and Chong Wang [10]. The way CTR addresses that limitation is by allowing the item latent factor to

be an offset from topic proportion. So, using this way, an offset may help explain, for instance, an article A is more important to researchers interested in machine learning than it is to those interested in social network analysis. Therefore, CTR allows the item latent factor to also account for user ratings.

Fundamentally, CTR incorporates techniques of both collaborative filterings based on latent factor models and content analysis based on probabilistic topic modeling. According to the CTR model, items are generated by a topic model while users are represented with topic interests [10]. Therefore, CTR is considered one of the excellent hybrid models which shows that the combination of the content modeling with the matrix factorization methods produces more promising results compared to traditional RS. The graphical model of CTR along with its algorithm is shown below. Note that, in machine learning, graphical models are used to represent a repetitive process of the probabilistic model. Essentially, they represent a factorization of the joint distribution of hidden and observed random variables. Nodes are random variables, plate boxes denote the "loop" with a variable shown in the bottom right corner of the plate representing its number of iterations, and edges mean that there is dependence between random variables in the generative process. Note that grey nodes represent observed variables while blank nodes are hidden variables. Figure *3.1* shows the graphical model of CTR. We assume *U* users and *J* items for the RS. The rating variable $r_{uj} \in \{0, 1\}$ denotes whether the user *u* likes item *j* or not. Also, note that $r_{uj} = 0$ can be interpreted in two ways: either user *u* is not interested in item *j* or user *u* does not know about article *j*. For each user, we try to recommend potentially interesting items that are rated yet by this user. Assuming that there is *K* topics $\boldsymbol{\beta} = \beta_{1:K}$ in the whole corpus of items, the graphical model, and the generative process of the CTR model are illustrated in Figure *3.1* and Algorithm *3.1*, respectively, below:
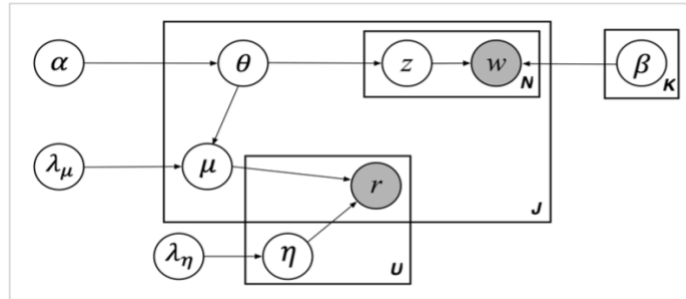

Figure 3.1: Graphical model for CTR.

1. For each user u, draw user latent vector $\eta_u \sim \mathcal{N}\left(0, \lambda_\eta^{-1} I_K\right)$.
2. For each item j,
   a. Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$.
   b. Draw item latent offset $\epsilon_j \sim \mathcal{N}\left(0, \lambda_\mu^{-1} I_K\right)$ and set the item latent vector as $\mu_j = \epsilon_j + \theta_j$.
   c. For the *n*-th word of item *j*,
      i. Draw topic index $z_{jn} \sim Mult(\theta_j)$.
      ii. Draw word $w_{jn} \sim Mult\left(\beta_{z_{jn}}\right)$.
3. For each user-item pair (u, j), draw the rating

$$r_{uj} \sim \mathcal{N}\left(\eta_u^T \mu_j c_{uj}^{-1}\right).$$

where c_{uj} is the confidence parameter for r_{uj}. For instance, we trust r_{uj} more if c_{uj} is large.

Algorithm 3.1: CTR model algorithm.

Despite its many advantages, the CTR model has significant computational limitations as well. The reason is that the model considers user ratings to have a Gaussian distribution which leads to iterating over all the entries in the rating matrix

during training. Because of this, CTR is highly inefficient considering that real-world datasets are very big and sparse. Additionally, CTR is a non-conjugate model [11], which makes it difficult to fit, challenging to work with on sparse data, and challenging to scale without stochastic optimization.

3.3 CTPF

To address CTR's inefficiency mentioned above, a newer hybrid model called Collaborative Topic Poisson Factorization (CTPF) has been proposed by Prem Gopalan, Laurent Charlin, and David M. Blei [12]. Fundamentally, CTPF incorporates concepts from two existing models: Poisson factorization [13] and CTR [10].

Poisson factorization substitutes a Poisson likelihood and non-negative representations for the conventional Gaussian likelihood and real-valued representations. In comparison with Gaussian factorization, Poisson factorization possesses more efficient inference, better handling of sparse data, and better predictive performance. So, the CTPF model assumes both reader behavior and item texts with Poisson distributions. As a result, CTPF is only concerned with non-zero ratings during training, and therefore it is much more efficient and scalable.

Compared to the CTR model, which is a non-conjugate model, CTPF is a conditionally conjugate model which allows us to use standard variational inference with closed-form updates. Moreover, CTPF, because it is based on Poisson and gamma variables, it has a more efficient and simpler-to-implement inference algorithm, and a much better fit to sparse real-world data. It is more scalable and provides significantly better recommendations than CTR [10].

We assume we have data containing U users and J items for the RS. CTPF assumes a collection of K unnormalized topics $\beta = \beta_{1:K}$. Each topic β_k is a collection of word intensities on a vocabulary of size V. Each unnormalized topic component β_{vk} is drawn from a Gamma distribution. CTPF assumes that, given the topics, a document j is generated with a vector of K latent topic intensities θ_j and that users are represented by a vector of K latent topic preferences η_u. In addition, the model assigns each document K latent topic offsets d that represent its deviation from the topic intensities. These deviations happen when a document's content does not sufficiently describe its ratings. Finally, CTPF claims that the conditional probability that a user u rated document j with rating r_{uj} is derived from a Poisson distribution with rate parameter $\eta^T_u(\theta_j + \epsilon_j)$ where ϵ_j is the document topic offset. The graphical model of CTPF along with its algorithm is demonstrated in Figure *3.1* and Algorithm *3.2*, respectively, below:

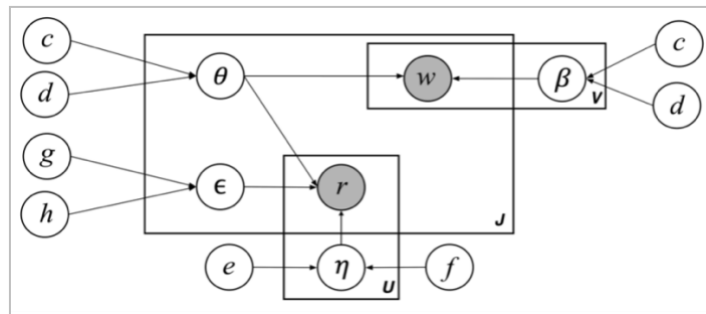Figure 3.2: Graphical model for CTPF.

1. **Item model:**
   a. Draw topics $\beta_{vk} \sim \text{Gamma}(a, b)$
   b. Draw item topic intensities $\theta_{jk} \sim \text{Gamma}(c, d)$
   c. Draw word count $w_{jv} \sim Poisson(\theta_j^T \beta_v)$.
2. **Recommendation model:**
   a. Draw user preferences $\eta_{uk} \sim \text{Gamma}(e, f)$
   b. Draw item topic offsets $\epsilon_{jk} \sim \text{Gamma}(e, f)$
   c. Draw $r_{uj} \sim \text{Poisson}\left(\eta_u^T(\theta_j + \epsilon_j)\right)$

Algorithm 3.2: Generative process for CTPF.

CTPF has two main advantages over previous work; having a conditionally conjugate model which helps to employ standard variational inference with closed-form updates and having built on Poisson factorization which makes the most use of sparsity of user consumption of items, therefore can analyze massive real-world data [12].

3.4 CTMP

Although all hybrid models mentioned above benefit from the interpretable semantics of the item latent factor, they still have some limits in terms of computational cost or predictive performance. Therefore, in this thesis, we will explore and implement a hybrid, scalable and interpretable probabilistic content-based collaborative filtering model called **Collaborative Topic Model for Poisson distributed ratings (CTMP)** [1]. CTMP covers the limitation of CTR by considering ratings in Poisson distribution as CTPF does, while modeling contents with LDA. Thus, it outperforms all previous hybrid models in terms of performance. Details of CTMP formalization, graphical model, and algorithm are discussed in section 7.

4 LDA

In machine learning, topic modeling is a statistical model for discovering a set of topics that occur in a collection of documents [14]. It is also considered a probabilistic model which offers an interpretable low-dimensional representation of the documents. For many years, the implementation of topic models for document classification, corpus exploration, and information retrieval has been of interest.

There are many topic modeling algorithms, among which LDA is the most popular one. LDA is a three-level hierarchical Bayesian model, and its basic idea is that documents are represented as random mixtures over an underlying set of topics, where each topic is characterized by a distribution over words that are biased around those associated under a single theme [5]. Therefore, topic probabilities express an explicit representation of each document. This can also be explained as below:

- *Each document is a mixture of topics:*
  Each document contains terms/words from some topics in specific proportions. For instance, if we consider that there are 2 topics in the whole corpus, then we might state that some documents could be 75% topic A, and 25% topic B, while another document could consist of 30% topic A, and 70% topic B.
- *Each topic is a mixture of words:*
  Each topic is expressed by the words that explain it most. For example, if we consider that there are 2 topics, namely, "sports" and "education", in the whole corpus, then the most used words for the sports topic could be "teammate", "win", and "play", while the education topic could contain the words such as "lecture", "book" and "class". It is necessary to note that the same words can appear on multiple topics. For example, the word "time" could participate in both sports and education topics.

In this way, documents can overlap with each other in their contents, rather than being separated into different individual groups. The generative process and graphical model of LDA for each document in the whole corpus are described below.

4.1 Learning

The terminology for the LDA model is as follows:

- *Word* is a term of the vocabulary, and it is indexed by $1, \ldots, V$.
- *Document* is a series of words given by $w = (w_1, w_2, \ldots, w_N)$, where w_n is the nth word inside the document.
- *Corpus* is a collection of a total J documents, and it is given by $D = (w_1, w_2, \ldots, w_J)$.
- K is the number of topics to be extracted from the corpus.
- α is Dirichlet prior parameter on per-document topic proportions.
- β is Dirichlet's prior parameter on per-topic word proportion.
- θ_j is topic proportions for document j.
- φ_k is word distribution for topic k.
- z_{jn} is the topic for an n-th word in document j.
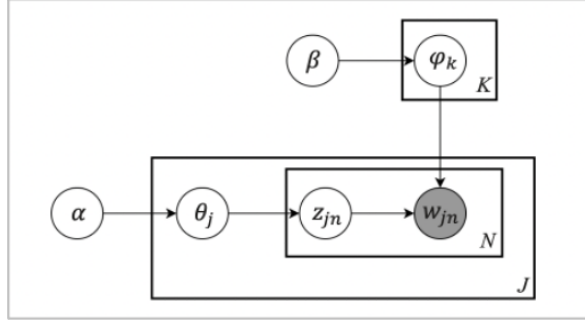
- w_{jn} is a specific word.

Figure 4.1: Graphical model for LDA.

Contrary to the original paper on LDA [5], a sparse Dirichlet prior can be employed to model the topic-word distribution, based on the idea that the probability distribution over words in a topic is skewed, with only a small subset of words having a high probability. This slightly updated model is the most extensively employed variation of LDA today. A graphical model of this slightly modified LDA is shown in Figure *4.1*.

It is also important to emphasize that the overall LDA process is a hidden generative process and according to this process, the model is assumed to generate the observed data (e.g., items, movies, documents). This was just a generative assumption to facilitate the algorithm and it does not illustrate the true process of the real data [15]. The following is how we view the generative process: documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over all the words. LDA assumes the following generative process for a corpus *D* consisting of *J* documents:

1. Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$, where $j \in \{1, \dots, J\}$.
2. Draw word distribution $\varphi_k \sim Dirichlet(\beta)$, where $k \in \{1, \dots, K\}$.
3. For the *n*-th word of document *j*,
   a. Draw topic index $z_{jn} \sim Mult(\theta_j)$.
   b. Draw word $w_{jn} \sim \text{Mult}\left(\varphi_{z_{jn}}\right)$.

Algorithm 4.1: Generative process for LDA.

As seen in Algorithm *4.1* above, the topics that the LDA algorithm tries to find from the whole corpus are treated as hidden variables. Each document of the corpus is represented in terms of topic proportions. Topic proportion θ_j is a *K*-dimensional Dirichlet random variable, and its domain is in the $(K-1)$-simplex. In other words, *K*-vector θ_j is in the $(K-1)$-simplex, therefore, $\theta_{ji} \geq 0$, $\sum_{i=1}^{K} \theta_{ji} = 1$. Also, φ_k is a *V*-dimensional Dirichlet random variable, and its domain is in $(V\text{-}1)$-simplex. Therefore, $\varphi_{ki} \geq 0$, $\sum_{i=1}^{V} \varphi_{ki} = 1$. Note that the Dirichlet is an exponential family distribution and one of its important properties is that it is conjugate prior to the multinomial distribution [16]. This conjugacy between these two distributions is important (refer to section 5.3.1.1.1), as it facilitates the development of the inference and parameter estimation for LDA.
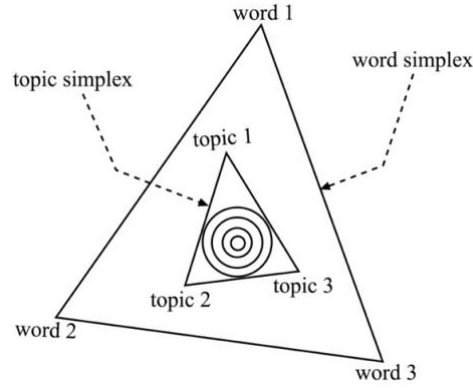
Figure 4.2: Topic simplex for 3 topics embedded in the word simplex for 3 words.

An example geometry of the latent space of LDA is illustrated in Figure *4.2* above.

4.2 Inference and Parameter Estimation

Computing the posterior distribution over the latent variables given some documents is the main inferential problem here because the posterior inference is intractable to compute:

$$p(\theta, \varphi, z \mid w, \alpha, \beta) = \frac{p(\theta, \varphi, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)} = \frac{p(\theta, \varphi, z, w \mid \alpha, \beta)}{\int_{\varphi} \int_{\theta} \sum_{z} p(\varphi, \theta, z, w \mid \alpha, \beta)} \qquad (4.1)$$

As normalization constant which depends on model parameters i.e., marginal probability $p(w \mid \alpha, \beta)$ contains intractable integrals in Eq. (4.1), the resulting posterior inference also becomes intractable to compute. Therefore, as an exact posterior distribution is not possible, several approximate inference algorithms can be used for LDA. For example, Variational Inference (VI) and relevant variational Expectation-Maximization (EM) algorithm can be used to learn the topics and decompose each document of the corpus according to these learned topics [5]. Details of VI are discussed in the following section.

5 Variational Inference

Variational Bayesian Methods (VBM) are a group of widely used techniques in the field of statistical Machine Learning. Suppose the following probabilistic model with the joint distribution of the observed variables *X* and the hidden variables *Z*:

$$p(Z, X)$$

Following the *Bayes' Theorem*, to infer the hidden variables *Z*, the posterior inference is used as follows:

$$\overbrace{p(Z|X)}^{posterior} = \frac{p(Z,X)}{p(X)} = \frac{p(X|Z)p(Z)}{p(X)} = \frac{\overbrace{p(X|Z)}^{likelihood}\ \overbrace{p(Z)}^{prior}}{\underbrace{\int_Z p(X,Z)}_{normalization\ constant}} \tag{5.1}$$

- Prior $p(Z)$ – is the probability of hidden variables before having seen any data. In other words, prior is the probability distribution which expresses one's beliefs about an event before some data is considered.
- Likelihood $p(X|Z)$ – is the probability of observed variables given hidden variables.
- Posterior $p(Z|X)$ – is the probability of hidden variables given the observed variables.
- Normalization constant $\int_Z p(X,Z)$ – is a marginal probability of observed variables or evidence, which does not depend on the hidden variables since it contains integral over all possible sets of hidden variables. It is called the normalization constant because it assures that posterior density integrates into one.

For many interesting models, the denominator is computationally intractable, mainly because of integrals. This means that the exact inference of the posterior is not possible. One possible approach is to perform an approximate posterior inference, which is what VI offers. The reason behind the popularity of VI methods is that they let us solve statistical inference problems as an optimization problem and solve for hidden parameters by maximizing their objective function. The most often used VI method is the Mean-field Variational Inference (MFVI) which will be discussed in section 5.3. But before this, let's explore the main idea behind VI and the forms of statistical models it is applied to.
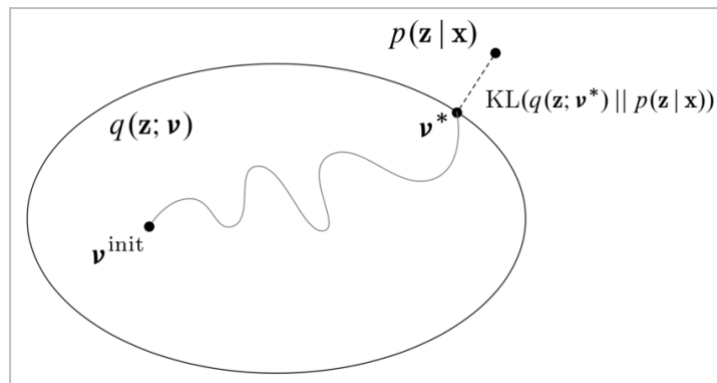

Figure 5.1: Approximate solution to the inference problem using Variational Inference (for picture, refer to [21]).

Figure *5.1* above simply illustrates a technique of VI [17]. Let's remember that VI aims to approximate the true posterior distribution $p(Z|X)$. To start with, one needs to posit a variational family of distribution over the hidden variables. This variational family is represented as an ellipse area in Figure *5.1*. As seen, it is also parametrized by variational parameters v. Next, the goal is to find v^* within this family of distributions, such that the corresponding approximate posterior distribution $q(Z; v)$ is closest to the true posterior distribution $p(Z|X)$. Note that this closeness is measured by Kullback-Leibler divergence (KL-divergence). Idea is to start at some initial set of variational parameters v^{init} and then optimize them - i.e., minimize KL divergence [18] to find the point where $q(Z; v)$ is closest to $p(Z|X)$.

5.1 KL–divergence derivation

$$KL\left[q(Z;v) \,||\, p(Z|X)\right]$$

$$= \int_Z q(Z;v) \log \frac{q(Z;v)}{p(Z|X)} = -\int_Z q(Z;v) \log \frac{p(Z|X)}{q(Z;v)}$$

$$= -\left(\int_Z q(Z;v) \log \frac{p(X,Z)}{q(Z;v)} - \int_Z q(Z;v) \log p(X) \right)$$

$$= -\underbrace{\int_Z q(Z;v) \log \frac{p(X,Z)}{q(Z;v)}}_{L} + \log p(X) \int_Z q(Z;v) \tag{5.2}$$

$$note: \int_Z q(Z;v) = 1 = -L + \log p(X)$$

where *L* above is called *the variational lower bound or evidence lower bound (ELBO)*. We reformulate the equation above as follows:

$$L = \log p(X) - KL\left[q(Z;v) \,||\, p(Z|X)\right] \tag{5.3}$$

Because the *KL* divergence is always non-negative (i.e., $KL[q(Z;v) \,||\, p(Z|X)] \geq 0$), we get $L \leq \log p(X)$. This proves that *L* is the lower bound on the log marginal probability of the observations. So, the final goal is to, using a coordinate ascent optimization algorithm (e.g., variational EM [19]), *maximize this lower bound L i.e., minimize KL divergence with respect to variational parameters v*. Note that $\log p(x)$ in the formula above is fixed against all variational parameters v.

5.2 Jensen's inequality derivation

Apart from the KL-divergence derivation mentioned above, there is also an alternative way to arrive at similar conclusions using Jensen's inequality. This is the most widely known ELBO derivation, which demonstrates why the ELBO is a lower bound on the evidence. It states $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ for the concave log function as follows:

$$\log p(X) = \log \int_Z p(X,Z)$$

$$= \log \int_Z p(X,Z) \frac{q(Z;v)}{q(Z;v)} = \log \left(\mathbb{E}_q \left[\frac{p(X,Z)}{q(Z;v)} \right] \right) \geq \underbrace{\mathbb{E}_q \left[log \frac{p(X,Z)}{q(Z;v)} \right]}_{L} \tag{5.4}$$

As shown in Eq. (5.4) above, the last term in the equation is the variational lower bound or ELBO. Essentially, it is again shown that L is the lower bound of the log marginal probability of the observations and our goal is to maximize the lower bound L where:

$$\log p(X) \geq L \tag{5.5}$$

5.3 Mean-field Variational Inference

Furthermore, if we consider MFVI, then the variational distribution over the hidden variables factorizes as follows:

$$q(Z; \nu) = \prod_{i=1}^{n} q(z_i; \nu_i) \tag{5.6}$$

The mean-field approximation makes a simplifying assumption by partitioning the hidden parameters into independent parts [20]. In other words, this assumption enforces full independence among all hidden parameters. The reason why this independence is very useful is that, when we use a coordinate ascent optimization algorithm such as variational EM, this assumption enables us to compute the update rules for each unknown parameter in isolation by keeping all others fixed [19].

5.3.1 Mean-field Variational Inference in conjugate models

Most importantly, it must be emphasized that there is a specific form for statistical models in which the coordinate ascent in MFVI yields closed-form updates. It is called exponential family conditionals or conditionally conjugate models. *Fundamentally, for a model to be conditionally conjugate, a complete conditional of each parameter must be in the exponential family and be in the same family as its prior* [12]. A complete conditional is the conditional probability of the hidden variable given all the observed variables and other hidden variables. A generic example to understand conditionally conjugate models is defined in Figure *5.2* below:
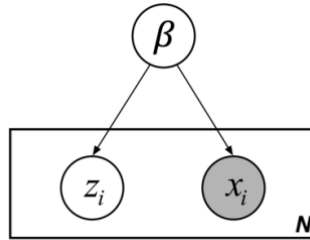

Figure 5.2: Graphical model for conditionally conjugate model (for picture, refer to [21]).

where $x = x_{1:N}$ are observed variables, $z = z_{1:N}$ are local hidden variables and $\beta = \beta_{1:N}$ are global hidden variables. Note that, the main difference between local and global hidden variables is that the i-th data only depends on global on β and z_i. In other words, it is not dependent on any other j-th local data point. Now, the factorized joint distribution of the model is as follows:

$$p(\beta, z, x) = p(\beta) \prod_{i=1}^{N} p(z_i, x_i | \beta) \tag{5.7}$$

and as usual, our goal is to compute a posterior $p(\beta, z|x)$.

Firstly, for the selected generic model above, the following complete conditionals must be in the exponential family:

$$p(z_i | \beta, x_i) = h(z_i) \, exp\{\eta_l(\beta, x_i)^T z_i - a(\eta_l(\beta, x_i))\}$$
$$p(\beta | \mathbf{z}, \mathbf{x}) = h(\beta) \, exp\{\eta_g(\mathbf{z}, \mathbf{x})^T \beta - a(\eta_g(\mathbf{z}, \mathbf{x}))\} \tag{5.8}$$

In mathematical terms, an exponential family is expressed as follows:

$$p(x) = h(x) \, exp\{\eta^T \, t(x) - a(\eta)\} \tag{5.9}$$

where η is a natural parameter, $t(x)$ is the sufficient statistics, $a(\eta)$ is the log normalizer, and $h(x)$ is the base density. In short, if some parameter x is in the exponential family, then it can be written in the form above. Secondly, the complete conditional must be in the same family as it is conjugate prior. Therefore, the exponential family was a crucial requirement in the first place. The reason is that for the exponential family of distributions, the likelihood distribution is a standardized function of the parameter, and therefore we can make conjugate priors by simulating the likelihood's form. Afterward, when a likelihood and a prior with the same exponential form are multiplied, the posterior maintains the same form, which was required as a second condition above. Fundamentally, an exponential family of distributions provides a beautiful theory around conjugate priors and corresponding posteriors and connects closely to variational inference [22]. Note that, examples for conjugate priors and their corresponding posteriors are shown in the next section below.

5.3.1.1 Conjugate Priors and Corresponding Posteriors

The main idea is that given a likelihood distribution, one needs to select a family of prior distributions such that computed posterior distribution is also included in this family. In this way, chosen conjugate prior enables us to estimate the posterior distribution just by updating the parameters of the prior distribution.

The exponential family of distributions is the best example of this. The Gaussian, beta, binomial, Dirichlet, multinomial, gamma, Poisson, exponential, geometric, categorical, chi-squared, and log-normal are all in the exponential family. Some pairs of conjugate distributions from the exponential family are shown below in detail.

5.3.1.1.1 Multinomial distribution and Dirichlet priors

Remember that the multinomial distribution is the probability distribution where outcomes from experiments are discrete and involve two or more variables. It is also considered a multivariate generalization of binomial distribution which involves only two outcomes. Mathematically, it is defined as follows:

$$p(\boldsymbol{x}|\theta) = \frac{(\sum_i x_i)!}{x_1! \, x_2! \ldots x_n!} \prod_{i=1}^{n} \theta_i^{x_i} \tag{5.10}$$

where, x_i indicates the number of times outcome i occurs out of n trials, while θ_i signifies the probability that outcome i occurs.

Now, let's remember the Dirichlet distribution which is a continuous multivariate probability distribution. It is also considered a multivariate generalization of beta distribution and is defined as follows:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} \tag{5.11}$$

where α is a k-vector with components $\alpha_i > 0$, θ is a k-dimensional random variable which is in $(k-1)$-simplex, therefore $\theta_i \geq 0$, $\sum_{i=1}^{n} \theta_i = 1$. Additionally, $\Gamma(\alpha)$ denotes the gamma function, where:

$$\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1} e^{-x} \, dx \tag{5.12}$$

According to conjugate Bayesian analysis, the Dirichlet distribution is considered a conjugate prior to the multinomial distribution. Therefore, when we multiply the likelihood expressed in multinomial form with the prior expressed in Dirichlet form, we get the posterior distribution as follows:

$$p(\theta|\boldsymbol{x}, \alpha) \propto P(\boldsymbol{x}|\theta) \, P(\theta|\alpha) = \frac{(\sum_i x_i)!}{x_1! \, x_2! \ldots x_n!} \prod_{i=1}^{n} \theta_i^{x_i} \frac{\Gamma(\sum_{i=1}^{n} \alpha_i)}{\prod_{i=1}^{n} \Gamma(\alpha_i)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}$$

$$\propto \frac{\Gamma(\sum_{i=1}^{n}(\alpha_i + x_i))}{\prod_{i=1}^{n} \Gamma(\alpha_i + x_i)} \prod_{i=1}^{n} \theta_i^{\alpha_i + x_i - 1} \tag{5.13}$$

which we can confirm that it has the form of Dirichlet distribution. As shown in Eq. (5.13) above, we can estimate the posterior distribution just by updating the parameters of the prior distribution. *Because Dirichlet is a conjugate prior for its multinomial distributed likelihood, it leads to the LDA model being a conditionally conjugate model, and therefore, having coordinate updates of MFVI in closed-form* [5].

5.3.1.1.2 Poisson distribution and gamma priors

Let's now consider the Poisson distribution from discrete exponential family distributions:

$$p(\boldsymbol{x}|\theta) = \frac{\theta^x e^{-\theta}}{x!} \tag{5.14}$$

where conjugate prior to this Poisson likelihood must also have the form of Poisson distribution:

$$p(\theta|\alpha) \propto \theta^{\alpha_1 - 1} e^{-\alpha_2 \theta} \tag{5.15}$$

This conjugate prior can be easily expressed as *gamma distribution*:

$$p(\theta|\alpha) = K(\alpha)\theta^{\alpha_1-1}e^{-\alpha_2\theta} \tag{5.16}$$

where,

$$K(\alpha) = \frac{\alpha_2^{\alpha_1}}{\Gamma(\alpha_1)} = \frac{\alpha_2^{\alpha_1}}{(\alpha-1)!} \tag{5.17}$$

$\Gamma(\cdot)$ denotes the gamma function above. Now, the prior-to-posterior update is as follows:

$$
\begin{aligned}
p(\theta|\boldsymbol{x},\alpha) \propto P(\boldsymbol{x}|\theta)P(\theta|\alpha) &= \prod_{x=1}^{n}\frac{\theta^x e^{-\theta}}{x!}\theta^{\alpha_1-1}e^{-\alpha_2\theta} \\
&= e^{-n\theta}\theta^{\sum_{i=1}^{n}x_i}\theta^{\alpha_1-1}e^{-\alpha_2\theta} \\
&= \theta^{\sum_{i=1}^{n}x_i+\alpha_1-1}e^{-\theta(n+\alpha_2)}
\end{aligned} \tag{5.18}
$$

where we can confirm that it has the form of gamma distribution which was our intention from the beginning. Essentially, as seen above, choosing Gamma conjugate prior and multiplying it to Poisson likelihood yielded the posterior inference which also has Gamma distribution, and therefore we can estimate the posterior distribution just by updating the parameters of the prior distribution, while successfully ignoring the intractable marginal probability in the denominator. In other words, if $x_1,\dots,x_n\sim Poisson(\theta)$ are all identically independently distributed, then conjugate prior for θ is Gamma(α_1,α_2) and the respective posterior, which is proportional to likelihood multiplied by prior becomes Gamma($\sum_{i=1}^{n}x_i+\alpha_1, n+\alpha_2$). *Because gamma is a conjugate prior to its Poisson distributed likelihood, it leads to the CTPF model being a conditionally conjugate model, and therefore, having coordinate updates of MFVI in closed-form* [12].

5.3.2 Mean-field Variational Inference in non-conjugate models

So far, from the previous sections, we have seen that if the model is conditionally conjugate, we can easily use MFVI to have a closed-form solution. Nevertheless, not all models are conditionally conjugate models; some are non-conjugate. In these models, the MFVI approach cannot be applied directly, and practitioners must create their case-specific variational algorithms. In section 7, we will see that the CTMP model is among these non-conjugate models where its authors developed a co-ordinate ascent algorithm to fit it [1].

6 OPE and BOPE

Maximum a posteriori probability (MAP) estimation has a significant impact on doing posterior inference (i.e., estimating hidden parameters) in many probabilistic models. Especially, many interesting MAP problems are continuous, non-convex, and intractable. In the field of non-convex optimization, there have been a variety of different techniques such as Frank–Wolfe [23], Natasha2 [24], Stochastic Majorization-Minimization [25], Concave-Convex procedure [26] which aim to solve the MAP problem. However, non-convex optimization is NP-hard, and the techniques mentioned above may not provide a viable solution for the MAP problem, because they disregard its special underlying structure. Therefore, for solving non-convex MAP problems with a state-of-the-art convergence rate, we will explore two efficient algorithms **Online Maximum a Posteriori Estimation (OPE)** [27] and its regularized, general, and more flexible version **Bernoulli randomness for Online maximum a Posteriori Estimation (BOPE***)* [28]. First, we introduce MAP estimation as the following task:

$$x^* = arg\ max_{x \in \Omega}\ P(x|D)$$
(6.1)

where we denote x as the hidden variable, D as the observed data, and Ω denotes $x's$ domain. Note that there also have been proposed many algorithms which directly try to estimate a full posterior distribution $P(x|D)$ mentioned above, i.e., Collapsed Gibbs Sampling (CGS) [29], Hessian Approximated Markov Chain Monte Carlo (HAMCMC) [30]. However, these methods provided suboptimal solutions along with a slow convergence rate. We continue by using *Bayes' Theorem*:

$$P(x|D) = \frac{P(D|x)P(x)}{P(D)} \propto P(D|x)P(x)$$
(6.2)

where we denote $P(D|x)$ as the likelihood of D, $P(x)$ as x's prior, and $P(D)$ as $D's$ marginal probability. Using Eq. (6.2) we rewrite Eq. (6.1) as follows:

$$x^* = arg\ max_{x \in \Omega}\ [f(x) = log\ P(D|x) + log\ P(x)]$$
(6.3)

We will focus on the conditions where the MAP problem is continuous and non-convex, hence intractable, i.e., $-f(x) = -log\ P(D|x) - log\ P(x)$ is non-convex over the continuous compact domain Ω [28]. As previously mentioned, MAP problem in Eq. (6.3) will be treated as an optimization problem. Therefore, objective function $f(x) = g_1(x) + g_2(x)$ defines the complexity of this optimization problem where $g_1(x) = log\ P(D|x)$ and $g_2(x) = log\ P(x)$. So, our problem in Eq. (6.3) becomes a non-convex constrained optimization problem as follows:

$$x^* = arg\text{max}_{x \in \Omega}\ [f(x) = g_1(x) + g_2(x)]$$
(6.4)

So, in the following sections, we will discuss OPE and BOPE algorithms for solving the optimization problem shown above.

6.1 OPE for solving MAP problem

OPE is considered a type of iterative optimization algorithm, which is the stochastic version of the Frank–Wolfe algorithm. The biggest advantage of OPE is that it has a faster convergence rate of $\mathcal{O}(1/T)$ to local maximal point compared to the existing stochastic algorithms for nonconvex problems, where T signifies the number of iterations during training of its following algorithm [27]:

Output: x^* which maximizes the objective function $f(x) = g_1(x) + g_2(x)$ over the compact domain Ω.
Initialize x_1 arbitrary in Ω.
1. **for** $t = 1, 2, \ldots, \infty$ **do**
2.   Pick f_t uniformly from $\{g_1(x), g_2(x)\}$
3. $F_t := \frac{2}{t} \sum_{h=1}^{t} f_h$
4. $a_t := arg\ max_{x \in \Omega} \langle F_t'(x_t), x \rangle$
5. $x_{t+1} := x_t + \frac{a_t - x_t}{t}$
6. **end for**

Algorithm 6.1: Online Maximum a Posteriori Estimation (OPE) algorithm.

As illustrated in Algorithm *6.1*, the OPE algorithm solves a linear program at each iteration, i.e., directing the optimization solution to the good vertex in the convex hull of the compact input domain. In more detail, what OPE does is to develop a sequence of stochastic functions $F_t(x)$ that approximates to $f(x)$ by alternatively selecting an f_t from $\{g_1(x), g_2(x)\}$ uniformly randomly at each iteration t. As proved in its original paper [27], $F_t(x)$ converges to f_t as. $t \rightarrow infinity$.

Despite the fast convergence rate, OPE still has a limitation. As stated in the algorithm, either likelihood $g_1(x)$ or prior $g_2(x)$ is being used during the construction of the approximation function $F_t(x)$. However, when dealing with new samples, we can rely on likelihood if we have seen enough data or rely on prior if there is a lack of data.

6.2 BOPE for solving MAP problem

To overcome the OPE's limitation mentioned above, a new approximation technique to OPE has been proposed as BOPE which retains all theoretical guarantees of OPE's convergence while being more general and flexible by using Bernoulli distribution and two stochastic bounds [28]. In general, both OPE, and BOPE try to lead the solution of the optimization to the closed neighbors of the vertices in the convex hull of the compact input domain and they have a fast convergence rate of $\Theta(1/T)$ along with proven quality bound [28]. BOPE solves the Eq. (6.4) above by employing Bernoulli distribution with parameter $p \in (0,1)$ which is supposed to replace the uniform distribution of OPE on likelihood and prior. Furthermore, as seen in Algorithm *6.2* below, during the procedure, two stochastic sequences are constructed and they converge to the objective function $f(x)$: the lower sequence L_t, and the upper sequence U_t. It is worth mentioning that the Bernoulli parameter p determines an impact of likelihood and prior on L_t and U_t. So, during each iteration, using both L_t and U_t stochastic sequences provide further information about $f(x)$, therefore increasing the chances of converging to $f(x)$ more quickly [28]. Both lower and upper sequences are guaranteed to converge to $f(x)$ as $t \rightarrow infinity$.

Input: Bernoulli parameter $p \in (0, 1)$

Output: x^* which maximizes $f(x) = \log P(D|x) + \log P(x)$ over the compact domain Ω.

1. **init** Initialize x_1 arbitrarily in Ω.

2. $G_1(x) := \frac{1}{p}\log P(D|x)$; $G_2(x) := \frac{1}{1-p}\log P(x)$

3. $f_1^l := G_1(x)$, $f_1^u := G_2(x)$

4. **for** $t = 1, 2, \ldots, T$ **do**

5.   Pick f_{t+1}^l randomly from $\{G_1(x), G_2(x)\}$ according to the Bernoulli distribution with parameter p, where
   $$P(f_{t+1}^l = G_1(x)) = p; \ P(f_{t+1}^l = G_2(x)) = 1 - p$$

6. $L_t := \frac{1}{t}\sum_{h=1}^{t} f_h^l$

7. $a_t^l := \arg\max_{x \in \Omega} < L_t'(x_t), x >$

8. $x_{t+1}^l := x_t + \frac{a_t^l - x_t}{t}$

9.   Pick f_{t+1}^u randomly from $\{G_1(x), G_2(x)\}$ according to the Bernoulli distribution with parameter p, where
   $$P(f_{t+1}^u = G_1(x)) = p; \ P(f_{t+1}^u = G_2(x)) = 1 - p$$

10. $U_t := \frac{1}{t}\sum_{h=1}^{t} f_h^u$

11. $a_t^u := \arg\max_{x \in \Omega} < U_t'(x_t), x >$

12. $x_{t+1}^u := x_t + \frac{a_t^u - x_t}{t}$

13. $x_{t+1} := \arg\max_{x \in \{x_{t+1}^l, x_{t+1}^u\}} f(x)$

14. **return** x_T

Algorithm 6.2: Bernoulli randomness for Online maximum a Posteriori Estimation (BOPE) algorithm.

It's important to note that one of the reasons why BOPE outperforms OPE is that we can create variants of BOPE by altering the Bernoulli parameter p. In addition to this, another property of BOPE is that to prevent overfitting of the learning process which is a widespread issue that affects all machine learning techniques, BOPE employs implicit regularization. Specifically, according to the original paper [28], Bernoulli randomness operates as a regularizer and BOPE uses an implicit prior that is stochastically vanishing with respect to iterations T. Note that this implicit prior is not the same as the prior used in MAP estimation. This implicit regularization is very critical, especially in RS where most of the datasets are *sparse* which makes the models prone to overfitting. Therefore, using BOPE instead of OPE in CTMP will facilitate the learning procedure and prevent overfitting.

7 Collaborative Topic Model for Poisson distributed ratings

In the following sections, we describe formalization, inference, learning parameters, prediction phases, and key properties of CTMP.

7.1 Formalization

Before diving into technical parts, let's provide some notations:

- *U:* represents the number of users inside the dataset.
- *J:* represents the number of items inside the dataset.
- $w_j = c_j^{v}{}_{v=1}^{V}$: describes the bag-of-word representation for each item j where c_j^v expresses the frequency of term/word v in item j.
- V: represents the vocabulary size of the corpus.
- $D = r_{uj}, w_j{}_{u=1, j=1}^{U, J}$: describes the dataset where r_{uj} is a rating provided by user u to item j, while w_j is the bag-of-word representation of item j as already explained above. $R = r_{uj}{}_{U \times J}$ represents the ratings given to movies by users. Every rating r_{uj} is expressed as binary 0 or 1. If user u liked an item j, then $r_{uj} = 1$. On the contrary, if the user u does not know about the item j or does not like it, then $r_{uj} = 0$.
- *K:* represents the number of topics inside the corpus.
- $\beta = \beta_{kv}{}_{K \times V}$: describes the topic representation. More precisely, every topic k is a distribution over the vocabulary. It is described as $\beta_k = \beta_{kv}{}_{V \times 1}$ where $\sum_{v=1}^{V} \beta_{kv} = 1$ and $\beta_{kv} \geq 0$. Note that, β_k lies in the $(k-1)$-simplex.
- $\theta_{1:J}$: describes the topic proportion of the items. $\theta_j = \{\theta_{jk}\}_{K \times 1}$ is the vector of the distribution on topics for item j, and $\sum_{k=1}^{K} \theta_{jk} = 1$, $\theta_{jk} \geq 0$. Note that, θ_j lies in the $(k-1)$-simplex.

To learn the topics $\beta = \beta_{kv}{}_{K \times V}$, we use the LDA and its EM approach which was described in the respective section of LDA. Furthermore, by learning the topic proportion of each item $\theta_j = \{\theta_{jk}\}_{K \times 1}$, we later describe each item and user in the K-dimensional space. Note that these learning procedures will be explained in section 7.3 below.

Now, we present latent factors for each user and item in terms of K-dimensional vectors $\eta_u = \eta_{uk}{}_{K \times 1}$ and $\mu_j = \mu_{jk}{}_{K \times 1}$, respectively. The reason why we consider μ_j rather than θ_j as the latent factor for an item is that to have a better recommendation system, we allowed an offset between μ_j and θ_j which accounts for the user-specific preference on the item content that θ_j alone can not capture. Therefore, we denote that $\mu_j = \theta_j + \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0, \lambda^{-1} I_K)$ is an offset term which has Gaussian distribution. Note that I_K in the formula above represents a K-dimensional identity matrix, and λ is a regularization parameter. So, we have $\mu_j \sim \mathcal{N}(\theta_j, \lambda^{-1} I_K)$.

Furthermore, the ratings and users' latent factors are modeled by Poisson and Gamma distributions, respectively. To put everything together, the generative

process and graphical model of CTMP are shown in Algorithm *7.1* and Figure *7.1* below.

1. For each user *u*, draw η_u where $\eta_{uk} \sim \text{Gamma}(e, f)$
2. For each item *j*:
    (a) Draw topic proportion $\theta_j \sim \text{Dirichlet}(\alpha)$
    (b) For the *n*-th word of item *j*:
        i. Draw topic index $z_{jn} \sim \text{Categorical}(\theta_j)$
        ii. Draw word $w_{jn} \sim \text{Categorical}(\beta_{z_{jn}})$
    (c) Draw latent factor $\mu_j \sim \mathcal{N}(\theta_j, \lambda^{-1} I_K)$
3. For each user-item pair (u, j), draw $r_{uj} \sim \text{Poission}(\eta_u^T \mu_j)$

Algorithm 7.1: Generative process for CTMP.

Note that steps 2(a-b) in Algorithm *7.1* correspond to LDA.
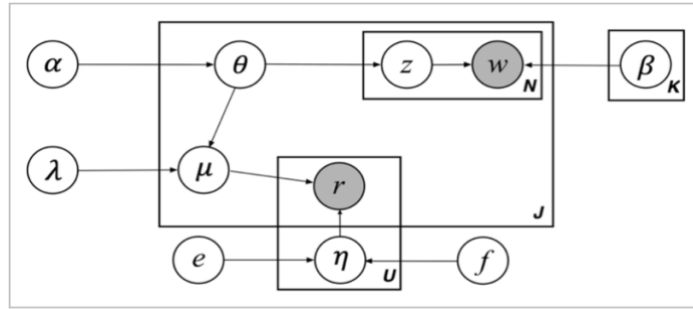
Figure 7.1: Graphical model for CTMP.

7.2 Inference

Full posterior of latent variables is given as follow:

$$P(\theta, \mu, \eta | D, \alpha, \beta, \lambda, e, f) = \frac{P(\theta, \mu, \eta, D | \alpha, \beta, \lambda, e, f)}{P(D | \alpha, \beta, \lambda, e, f)} \quad (7.1)$$

The problem with this posterior is that it is intractable, and therefore exact inference is impossible. To tackle this problem, we have two methods:

1) MAP for point estimation
2) Bayesian Learning such as MCMC Sampling or Variational Methods for approximate inference

As the prior and posterior distributions of hidden variables, θ and μ are not conjugate in the CTMP model, using Variational Inference Methods to infer these hidden variables does not get us a closed-form solution. Therefore, we will carry out the point estimates of θ_j and μ_j using the MAP – coordinate ascent algorithm developed by the authors of the original paper of CTMP [1].

Furthermore, to facilitate the learning, the authors added a new auxiliary variable *y*, where $y_{ujk} \sim \text{Poisson}(\eta_{uk}\mu_{jk})$ and $r_{uj} = \sum_{k=1}^{K} y_{ujk}$. Note that we approximate the posterior of η_u and y_{uj} via MFVI [12]. MFVI is a type of VBM which allows to re-write a statistical inference problem as an optimization problem [8]. Therefore, we can convert the inference problem of CTMP into a full optimization problem where the single objective function which needs to be maximized is as follow:

$$L = \log P(\theta, \mu, D | \alpha, \beta, \lambda, e, f)$$

$$= \sum_{j=1}^{J} \log P(\theta_j, \mu_j, w_j | \alpha, \beta) + \sum_{u=1}^{U} \sum_{j=1}^{J} \log P(r_{uj} | \mu_j, e, f)$$

$$= \sum_{j=1}^{J} \log P(\theta_j, w_j | \alpha, \beta) + \sum_{j=1}^{J} \log P(\mu_j | \theta_j, \lambda) \qquad (7.1)$$

$$+ \sum_{u=1}^{U} \sum_{j=1}^{J} \log \int \sum_{y_{uj}} P(r_{uj}, y_{uj}, \eta_u | \mu_j, e, f) d\eta_u$$

As shown in Eq. (7.1), the term integration and summation over the whole space causes optimization to be intractable. However, the Variational method [31] also tackles this problem which will be discussed in detail below.

Note that y_{ujk} has Poisson distribution, and the K-dimensional vector y_{uj} follows multinomial distribution: $Mult\left(r_{uj} | \pi_{ui} = \left\{\frac{\eta_{uk}\mu_{jk}}{\eta_u^T \mu_j}\right\}\right)$ [32]. So, we get the variational distribution as follows:

$$q(\eta_u, y_{uj}) = q(y_{uj} | r_{uj}, \phi_{uj}) \prod_{k=1}^{K} q(\eta_{uk} | shp_{uk}, rte_{uk}) \qquad (7.2)$$

where,

$$q(y_{uj} | r, \phi_{uj}) \stackrel{def}{=} Mult(y_{uj} | r, \phi_{uj})$$

$$q(\eta_{uk} | shp_{uk}, rte_{uk}) = Gamma(\eta_{uk} | shp_{uk}, rte_{uk}) \qquad (7.3)$$

$$\phi_{uj} = \phi_{ujk_{K \times 1}}$$

such that ϕ_{uj} is a variational parameter of y_{uj}, and (shp_{uk}, rte_{uk}) are variational parameters of η_u. Note that $\sum_{k=1}^{K} \phi_{ujk} = 1$. Now we get the ELBO ($l$) by applying Jensen's inequality:

$$L = \sum_{j}^{J} \log P(\theta_j, w_j | \alpha, \beta) + \sum_{j}^{J} \log P(\mu_j | \theta_j, \lambda)$$

$$+ \sum_{u}^{U} \sum_{j}^{J} \log \int \sum_{y_{uj}} \frac{P(r_{uj}, y_{uj}, \eta_u | \mu_j, e, f)}{q(y_{uj}, \eta_u)} q(y_{uj}, \eta_u) d\eta_u$$

$$\geq \sum_{j}^{J} \log P(\theta_j | \alpha, \beta, w_j) + \sum_{j}^{J} \log P(w_j | \theta_j, \lambda)$$

$$+ \sum_{u}^{U} \sum_{j}^{J} (\int \sum_{y_{uj}} q(y_{uj}, \eta_u) \log \frac{P(r_{uj}, y_{uj}, \eta_u | \mu_j, e, f)}{q(y_{uj}, \eta_u)} d\eta_u) \qquad (7.4)$$

$$= \sum_{j}^{J} \log P(\theta_j, w_j | \alpha, \beta) + \sum_{j}^{J} \log P(\mu_j | \theta_j, \lambda) \psi$$

$$+ \sum_{u}^{U} \sum_{j}^{J} (E_{q(y_{uj}, \eta_u)} \log P(r_{uj}, y_{uj}, \eta_u | \mu_j, e, f) - E_{q(y_{uj}, \eta_u)} \log q(y_{uj}, \eta_u)) = l$$

Note that before learning the hidden parameters, α, λ, e, f and K are considered fixed parameters in the model.

Next, the lower bound $l(\theta, \mu, \phi, shp, rte, \beta)$ is maximized with respect to $\theta, \mu, \phi,$ *shp, rte, β.* According to Appendix A**. Objective function** of the original CTMP paper [1], the terms are expressed in detail as follows:

$$l(\theta, \mu, \phi, shp, rte, \beta)$$
$$= \sum_j^J \left((\alpha - 1) \sum_{k=1}^K \log \theta_{jk} + \sum_v^V c_j^v \log \sum_k^K \theta_{jk} \beta_{kv} \right)$$
$$- \sum_j^J \frac{\lambda}{2} \|\theta_j - \mu_j\|_2^2 + \sum_u^U \sum_j^J \sum_k^K r_{uj} \phi_{ujk} \log(\mu_{jk}) - \sum_u^U \sum_j^J \sum_k^K r_{uj} \phi_{ujk} \log(\phi_{ujk})$$
$$+ \sum_u^U \sum_k^K \left(rte_{uk} - f - \sum_j^J \mu_{jk} \right) \frac{shp_{uk}}{rte_{uk}}$$
$$+ \sum_u^U \sum_k^K \left(\sum_j^J r_{uj}\phi_{ujk} + e - shp_{uk} \right) \left(\Psi(shp_{uk}) - \log(rte_{uk}) \right)$$
$$- \sum_u^U \sum_k^K shp_{uk} \log(rte_{uk}) + \sum_u^U \sum_k^K \log\left(\Gamma(shp_{uk})\right) + Constant$$

(7.5)

7.3 Learning Parameters

Eq. (7.5) above is the optimization problem and as mentioned before we solve it by coordinate ascent algorithm. CTMP algorithm for learning $\theta, \mu, \phi, shp, rte$ and β is demonstrated in Algorithm *7.2* below:

Input: Observed data *w, r*, Bernoulli parameter $p \in (0,1)$ *and hyper − parameters α, λ, e, f.*
Output: Estimates $\theta, \mu, \phi_{uj}, shp_{uk}, rte_{uk}$ and β.
1. **init** Initialize θ, β *by their respective estimates from LDA*
2. **repeat**
3.   **for** $j = 1 : J$ **do**
4.   *Update θ_j by BOPE algorithm*
5.   *Update μ_j as in Equation 7*
6.   **end for**
7.   **for** $u = 1 : U, \ k = 1 : K$ **do**
8. $\phi_{uj} \propto exp\{\log \mu_{jk} + \psi(shp_{uk}) - log(rte_{uk})\} \forall j \ if \ r_{uj} > 0$
9. $shp_{uk} \leftarrow e + \sum_j r_{uj}\phi_{uj}$
10. $rte_{uk} \leftarrow f + \sum_j \mu_{jk}$
12.   **end for**
13. $\beta_{kv} \propto \sum_j c_j^v \theta_{jk}, \forall k, v$
14. **until** *convergence*

Algorithm 7.2: CTMP model algorithm.

Learning θ_j. To find the point estimate of local topic proportion θ_j, where

$$g(\theta_j) = (\alpha - 1) \sum_k \log \theta_{jk} + \sum_v c_j^v \log \left(\sum_k \theta_{jk} \beta_{kv} \right) - \frac{\lambda}{2} \|\theta_j - \mu_j\|_2^2$$

(7.6)

we use BOPE algorithm [28]. Note that in the original paper of CTMP, the authors have used a simple OPE algorithm. So, using BOPE instead of OPE to learn the topic proportions is the most important difference between our implementation of CTMP and the one in the original CTMP paper [1]. Let's remember that, by using Bernoulli randomness, BOPE achieves a faster convergence, and is more general and flexible compared to OPE. Furthermore, BOPE implicitly utilizes a prior which plays a regularization role [28]. Moreover, as mentioned earlier too, every topic proportion θ_j holds $\sum_{k=1}^{K} \theta_{jk} = 1$, $\theta_{jk} \geq 0$ and it lies in the $(k-1)$-simplex. BOPE algorithm for learning θ_j is described in Algorithm *7.3* below:

Input: Bernoulli parameter $p \in (0,1)$, $w_j = \{c_j^v\}_{v=1}^{V}$, λ, β, μ_j, α

Output: θ_j which maximizes $g(\theta_j)$ over the compact domain $\bar{\Delta}_K = \{x \in \mathbb{R}^K : \sum_k x_k = 1, x_k \geq \epsilon > 0\}$

1. **init** Initialize $\theta_{j(1)}$ arbitrarily in $\bar{\Delta}_K$

2. $G_1(\theta_j) := \dfrac{1}{p}\left((\alpha - 1)\sum_k \log \theta_{jk} + \sum_v c_j^v \left(\log \sum_k \theta_{jk}\beta_{kv}\right)\right)$; $\quad G_2(\theta_j) := \dfrac{1}{1-p}\left(-\dfrac{\lambda}{2}\|\theta_j - \mu_j\|_2^2\right)$

3. $f_{(1)}^l := G_1(\theta_j)$, $f_{(1)}^u := G_2(\theta_j)$

4. **for** $t = 1, 2, \ldots, T$ **do**

5. $\quad$ Pick $f_{(t+1)}^l$ randomly from $\{G_1(\theta_j), G_2(\theta_j)\}$ according to the Bernoulli distribution with parameter p, where,

    $\quad P(f_{(t+1)}^l = G_1(\theta_j)) = p$; $P(f_{(t+1)}^l = G_2(\theta_j)) = 1 - p$

6. $\quad L_t := \dfrac{1}{t}\sum_{h=1}^{t} f_{(h)}^l$

7. $\quad a_{(t)}^l := arg\,max_{\theta_j \in \bar{\Delta}_K} < L_t'(\theta_{j(t)}), \theta_j >$

8. $\quad \theta_j{}_{(t+1)}^l := \theta_{j(t)} + \dfrac{a_{(t)}^l - \theta_{j(t)}}{t}$

9. $\quad$ Pick $f_{(t+1)}^u$ randomly from $\{G_1(\theta_j), G_2(\theta_j)\}$ according to the Bernoulli distribution with parameter p, where

    $\quad P(f_{(t+1)}^u = G_1(\theta_j)) = p$; $P(f_{(t+1)}^u = G_2(\theta_j)) = 1 - p$

10. $\quad U_t := \dfrac{1}{t}\sum_{h=1}^{t} f_{(h)}^u$

11. $\quad a_{(t)}^u := arg\,max_{\theta_j \in \bar{\Delta}_K} < U_t'(\theta_{j(t)}), \theta_j >$

12. $\quad \theta_j{}_{(t+1)}^u := \theta_{j(t)} + \dfrac{a_{(t)}^u - \theta_j}{t}$

13. $\quad \theta_{j(t+1)} := arg\,max_{\theta_j \in \{\theta_j{}_{(t+1)}^l, \theta_j{}_{(t+1)}^u\}} f(\theta_j)$

Algorithm 7.3: Learning θ_j using BOPE.

Learning μ_j. If we know the estimates of other hidden variables, then solving μ_j analytically is possible because the objective function regarding the μ_j is concave.

$$f(\mu_j) = -\frac{\lambda}{2}\|\theta_j - \mu_j\|_2^2 + \sum_{u,k} r_{uj}\phi_{ujk}\log\mu_{jk} - \sum_k \mu_{jk}\sum_u \frac{shp_{uk}}{rte_{uk}} \tag{7.7}$$

The partial derivative of function $f(\mu_j)$ with respect to μ_j, i.e., $\dfrac{\partial f}{\partial \mu_{jk}}$ for all *k,* is the estimate of μ_j. This is also so-called the stationary point of $f(\mu_j)$. Because $\dfrac{\partial f}{\partial \mu_{jk}}$ is the quadratic function in terms of μ_{jk}, we can utilize Vieta's formula for the analytical derivation of the function's root as follows:

$$\mu_{jk} = \frac{-\sum_u \frac{shp_{uk}}{rte_{uk}} + \lambda\theta_{jk} + \sqrt{\Delta}}{2\lambda} \tag{7.8}$$

where,

$$\Delta = \left(-\sum_u \frac{shp_{uk}}{rte_{uk}} + \lambda\theta_{jk}\right)^2 + 4\lambda\sum_u r_{uj}\,\phi_{ujk} \tag{7.9}$$

Learning $\phi_{uj}, shp_{uk}, rte_{uk}$. We use MFVI for approximating the conditional posterior of η_u and y_{uj} as in [12]. So, to solve for the variational parameters of η_u and y_{uj} which are ϕ_{uj}, shp_{uk} and rte_{uk}, we solve for the stationary point of $log\,q(\eta_u, y_{uj})$ with respect to each variational parameter while holding the others same. The expression of the update of variational parameters is given in Eq. (7.10) below. The detailed derivation of these expressions is described in Appendix B**. Update $\boldsymbol{\phi_{uj}}$** and

Appendix **C. Update $\mathbf{shp}_{u,k}$ and $\mathbf{rte}_{uk}$**. One of the biggest advantages of the CTMP algorithm is that whenever $r_{uj} = 0$, we get $y_{ujk} = 0$ and $\phi_{ujk} = 0$ ($k \in \{1, \ldots, K\}$), and therefore, we only have to update ϕ_{ujk} over non-zero ratings ($r_{uj} > 0$). This property of our model diminishes the training time significantly, so the total training time is much lower than of other models such as CTR, especially whenever the rating dataset is highly sparse. Because, during each epoch of training, we only consider the positive ratings for updating the expression of ϕ_{ujk} and skip all *zero* ratings.

$$\begin{aligned}
\boldsymbol{y_{ujk}}: \quad & \phi_{uj} \propto \exp\left\{\log\mu_{jk} + \psi(shp_{uk}) - \log(rte_{uk})\right\} \\
\boldsymbol{\eta_{uk}}: \quad & shp_{uk} \leftarrow e + \sum_j r_{uj}\phi_{uj} \\
& rte_{uk} \leftarrow f + \sum_j \mu_{jk}
\end{aligned} \tag{7.10}$$

Note that the $\psi(\cdot)$ function in the update expression of ϕ_{ujk} denotes the digamma function:

$$\psi(x) := \frac{d}{dx}log\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)} \tag{7.11}$$

where $\Gamma(x)$ denotes the gamma function.

Learning β. So far, we have provided the update expression of the variables regarding both documents and users such as $\theta, \mu, \phi_{uj}, shp_{uk}$ and rte_{uk}. Now, we must do the remaining task which is to solve for β. First, we express the log-likelihood of the items' corpus C as in [33]:

$$Log\ P(C) = \sum_{j \in C} log\ P(j) = \sum_{j \in C} \sum_{v \in I_j} c_j^v\ log \sum_{k=1}^{K} \theta_{dk}\ \beta_{kv}$$

$$\geq \sum_{j \in C} \sum_{v \in I_j} c_j^v \sum_{k=1}^{K} \theta_{dk}\ log\ \beta_{kv} \tag{7.12}$$

By using Jensen's inequality, the last term is derived, because $\sum_k \theta_{jk} = 1$, $\theta_{jk} \geq 0, \forall k, j$. Next, the lower bound of $Log\ P(C)$ is maximized with respect to β as in [33]:

$$f(\beta) = \sum_{j \in C} \sum_{v \in I_d} c_j^v \sum_{k=1}^{K} \theta_{jk}\ log \beta_{kv} \tag{7.13}$$

where $\sum_{v=1}^{V} \beta_{kv} = 1, \beta_{kv} \geq 0, \forall k, v$. Note that the each β_k is separable from each other inside the objective function of $f(\beta)$. So, we can solve solve each β_k individually. This is carried out by considering the Lagrange function and setting its derivatives to 0 which results in the formula of β_{kv} as follows:

$$\beta_{kv} \propto \sum_{j \in C} c_j^v \theta_{jk} \tag{7.14}$$

7.4 Prediction

We rank the items to generate recommendations for each user u based on their predictive score s_{uj} after we have learned all the parameters. Because the ratings in the dataset are discrete Poisson variables, s_{uj} can be the expectation of the rate parameter given the observed data i.e., $\mathbb{E}[\eta_u^T \mu_j | D]$ as in CTPF [12]. However, the derivation in CTMP is a bit different because CTMP neither aims to approximate s_{uj} solely by point estimate nor require *conjugacy* between the complete *conditional distributions* for the inference as CTPF does [1]:

$$s_{uj} = \mathbb{E}[\eta_u^T \mu_j | D] \approx \mathbb{E}[\eta_u^T | D, \mu_j] \cdot \mu_j \tag{7.15}$$

Note that only μ_j is the MAP estimation of the complete conditional distribution. Furthermore, $\mathbb{E}[\eta_u^T | D, \mu_j]$ is nearly the expectation over the respective variational distributions of η_u^T's:

$$\mathbb{E}[\eta_u^T | D, \mu_j] \approx \mathbb{E}_{q(\eta_u^T | shp_{uk}, rte_{uk})}[\eta_u^T] = \frac{shp_{uk}}{rte_{uk}} \tag{7.16}$$

Note that both shp_{uk} and rte_{uk} are the estimation of variational parameters that we learned in section 7.3.

7.5 Key properties

Many key properties make CTMP perform much better than previous approaches in the RS field. To begin with, introducing LDA into the content production process in CTMP makes it more versatile than using Gamma mixtures in CTPF. This allows each item to discuss many topics. Secondly, by assuming the ratings to be Poisson distributed, all zero entries can be discarded in the estimation of the user latent factor η during the training phase.

Moreover, our empirical study also shows strong evidence that sparsity in the estimates of topic mixture can be recovered via learning. Note that sparsity is a highly important property since it facilitates the effective storage of data by providing a concise content representation. This allows an efficient computation of the tasks in industrial settings - for example, near-real-time product recommendations based on the same topics in which the consumer is interest.

8   Empirical Studies

We carry out empirical studies on the performance of the CTMP algorithm using a tech stack of Python, SQL, and Git. The personal repository which contains all the source code can be accessed via ***this link***.

For an empirical study of the CTMP algorithm under a real-world recommendation context, we use two different datasets, namely **MovieLens 20M** and **NETFLIX**. They are well-known datasets that have been used a lot and considered stable benchmark datasets for research purposes. Note that, for our empirical study, we use slightly modified versions of these datasets compared to the original versions. Their modified versions are arranged and put on Oracle Database by Michal Kopecký.

The original MovieLens 20M dataset describes 5-star rating activity from https://www.movielens.org which is a movie recommendation service. It was created on October 17, 2016, and its raw data consists of movies, users, ratings, and tag applications [34]. Each user is only represented by an id, therefore no other information (e.g., demographic) is considered. This dataset's modified version is arranged by M. Kopecký and it is also equipped with additional "TT" identifiers which help us to fetch the plots movies from the IMDB table. These plots are an essential part of our modified dataset because we will use the plots instead of the tag applications for the document representation. This is the important difference in empirical studies between our work and the original CTMP paper, and because of this, we will see slight differences in results too.

The original NETFLIX dataset was made available by the Netflix company for the competition that was held on September 21, 2009, for the best collaborative filtering algorithm to predict user ratings for movies, based on previous ratings without any other information about the users [35]. In other words, the users were not identified, they were only represented by id numbers for the contest. After the competition, the dataset became open-sourced and has since been used by many researchers as a stable benchmark dataset. Along with MovieLens 20M dataset, this NETFLIX dataset's modified version is also arranged by M. Kopecký and it is also equipped with additional "TT" identifiers which help us to fetch the plots movies from the IMDB table. These plots are an essential part of our modified dataset because we will use the plots instead of the tag applications for the document representation. This is the important difference in empirical studies between our work and the original CTMP paper, and because of this, we will see slight differences in results too.

8.1 Data preprocessing

After fetching both datasets from the Oracle database, we continue with data pre-processing steps for transforming the raw data into a useful and efficient format. By utilizing the data wrangling techniques, we make the individual data frames out of users, movies, and ratings as shown in Figure *8.1* and Figure *8.2* below:

Figure 8.1: Data frames for MovieLens 20M dataset.

Figure 8.2: Data frames for NETFLIX dataset.

Note that constructed data frames are still raw, meaning that ids in *df_user* data frame are not sequentially consistent, some movies in *df_movie* data frame do not have plots (i.e., filled with N/A values) and ratings in *df_rating* data frame are on the 0.5-5.0 scale. So, the next section uses these raw datasets for turning them into useful formats.

First, we start the cleaning process by representing ratings in binary form in ratings table. This is done by converting ratings bigger or equal to 4 into 1 and the remaining into 0. This way, it is presumed that users like or dislike the given movie. As previously stated, we do this because the CTMP model implies that ratings are Poisson distributed.

Next, we drop movies that have N/A plot. We also drop duplicates and inconsistent movies. Note that dropping movies results in the removal of some rows related to these movies in the ratings data frame. The size of each data frame before and after preprocessing is shown Table *8.1* below:

Data frame	Dataset	Raw rows size	Final rows size
df_user	MovieLens 20M	138,493	138,493
	NETFLIX	479,870	479,870
df_movie	MovieLens 20M	27,278	25,900
	NETFLIX	9,324	7,882
df_rating	MovieLens 20M	20,000,263	19,994,181
	NETFLIX	90,217,939	82,725,788

Table 8.1: Details of data frames.

8.1.1 Vocabulary Extraction

We now aim to extract a vocabulary separately for MovieLens 20M and NETFLIX using movie plots in the respective *df_movie* data frame. The reason why we want to have vocabulary is that each movie should be numerically represented for our Python model, therefore the vocabulary will be used for building this representation. Before we begin, we merge each movie plot sequentially to get a single long text. Then we extract the vocabulary from this single text. Note that, extracting the vocabulary requires careful investigation, because, for each distinct word, we should decide whether this word should be included in the vocabulary or not. Below are the steps that we follow for vocabulary extraction:

- Removing stop words – they are commonly used words (such as "a", "an", "are", "the", and "about") of a language that do not add much meaning to a sentence. They can be safely ignored by keeping them out of vocabulary. Many Python libraries provide a list of stop words for many languages. We used NLTK Library's stop words in the English language.
- Removing words of length less than 3 – these words can also be ignored. In the English language, there rarely is a word of length less than 3 which adds meaning to a sentence.
- Removing words with numbers or underscores – we will not include these words in the vocabulary.

It is also worth mentioning that in the domain of Natural Language Processing (NLP), well-known techniques such as Stemming, and Lemmatization are utilized for a purpose of text normalization. These techniques help a lot when it comes to filtering unwanted words during vocabulary construction. The stemming technique removes the suffix from a word and therefore reduces it to its root/stem. For example, a stem of the word "count" is just "count". Stemming the words such as "counts", "counting" and "counted" will result in "count". In contrast to stemming, lemmatization goes beyond word reduction by evaluating a language's whole lexicon for applying a morphological analysis to words. In other words, the lemmatization technique does not simply chop off inflections as stemming does, but instead it relies on a lexical knowledge base for obtaining the correct base forms of words. For example, given the word "mice", "ran" lemmatization converts them into "mouse" and "run", respectively. Essentially, by using stemming and lemmatization techniques, one can construct a more concise and accurate vocabulary. However, we don't use these techniques during our vocabulary construction. The reason behind this is that, according to recent works, stemming is claimed to reduce the model fit and negligibly affect topic coherence [36]. The utility of lemmatization on topic models is also vague and rather needs further investigation. Therefore, we avoid stemming and lemmatization of the corpus as a data pre-processing step.

Figure 8.3: Snippet from vocabulary extracted out of MovieLens 20M dataset.

Figure *8.3* shown above is a snippet from the resulting vocabulary of the MovieLens 20M dataset. We observe that the word "accumulate" has many forms with different suffixes. This is because we did not use stemming and lemmatization. We believe that letting vocabulary be rich like this will later facilitate model fit into the corpus. Overall, after extraction, MovieLens 20M vocabulary contains 59,110 words, while the NETFLIX dataset vocabulary has 34,177 words.

8.1.2 Movie Representation

After extracting vocabulary from the corpus, we use it for numerical representation of each movie/plot. The reason for this is that movies should be in such form that models (e.g., LDA, CTMP) can work with. Therefore, one way is to succinctly represent each movie as a sparse vector of word counts as below:

[M] [word_1]:[count] [word_2]:[count] ... [word_N]:[count]

where [M] denotes the number of unique words in the movie plot. Furthermore, [word_i] is an integer and it denotes an index of the word inside vocabulary, and [count] associated with this word denotes how many times the word appears in the movie plot. The snippet from numerical representation of movies is shown in Figure *8.4*. Note that if any word of the plot does not appear in corpus vocabulary, then we disregard it in movie representation.

Figure 8.4: Snippet from numerical representation of movies for MovieLens 20M dataset.

It is critical to remember that, the authors of both the original CTMP paper [1] and BOPE paper [28] implemented their models using tags for movie representation

rather than plots. This is the most significant distinction between our work and theirs. During the evaluation of the results, we will compare our model's output against theirs and determine whether using the movie plots resulted in different predictive performances.

8.1.3 Memory usage reduction

This technique helps us to reduce the memory usage of data frames. It iterates through all integer or float columns of the given data frame and modifies the data type to reduce memory usage. For instance, let's assume an integer column is given as an int64 data type. The way the function decided whether to reduce the memory or not is by first getting minimum and maximum values in this column. Next, it checks whether these minimum and maximum values are in the range of int32, int16, or int8 datatypes, and if they are, then it will convert the column's int64 datatype into the respective lower datatype. The same procedure applies to float columns as well. As proof, when we used this technique to reduce the memory storage of the *df_rating* data frame of the MovieLens 20M dataset, it achieved a 58.3% reduction in memory.

8.2 Model Fit

After preparing the corpus vocabulary, converting ratings into binary form, numerically representing movies/plots, and using the memory reduction technique on all data frames, we now continue with running and fitting the CTMP model using these data frames. Note that we will have separate models for MovieLens 20M and NETFLIX. For the model fitting phase, we will use vocabulary, numerical movie representations, and settings files.

We use *stratified 5-fold cross-validation* of ratings for training and testing the model. Generally, k-fold cross-validation is a widely used technique that assesses the efficacy of machine learning models since it produces a less biased estimate of their effectiveness. The parameter *k* indicates the number of folds that a given dataset is to split into. Pseudo-algorithm for model fitting with k-fold cross-validation is shown in Algorithm *3.1* below:

For each unique fold:
  1. Take this fold as test data.
  2. Take the remaining folds as training data.
  3. Fit a model on the training data and evaluate its performance on the test data.
  4. Keep the evaluation score and start from 1. using another unique fold.

Algorithm 4: Running the model with k-fold cross-validation.

At the end of the entire train-test circle, we average the evaluation scores kept for each fold to obtain a final evaluation score. For our case, we utilize stratified k-fold cross-validation, which has the same purpose as standard k-fold cross-validation but produces stratified folds which are made by preserving the percentage of samples for each class. This way, both train and test folds will contain information about each user, ensuring that the model will be trained and tested with all users. A visual example of it is shown in Figure *8.5* below:

Figure 8.5 Visual illustration of stratified 5-fold cross-validation on example dataset.

8.2.1 Hyperparameters

Along with vocabulary, and numerical movie representations files, we also have a settings file containing the model parameters. It helps us to set the model's parameters on a per-run basis. Example snippet of settings files and their content are shown in Figure *8.6* and. Table *8.2*, respectively, below:

```
1    num_movies: 25900
2    num_words: 59910
3    user_size: 138493
4    K: 100
5    tops: 10
6    lamb: 1
7    e: 0.3
8    f: 0.3
9    alpha: 1
10   p: 0.9
11   iter_infer: 50
12   iter_train: 50
```

Figure 8.6: Snippet from settings for CTMP input parameters for MovieLens 20M dataset.

Parameter	Description
num_movies	Number of documents in the corpus
num_words	Number of words of vocabulary
user_size	Number of users in dataset
K	Number of topics we want to discover from corpus
tops	Number of top words to extract from each topic for analyzing the quality of topics learned by the model
lamb (λ)	Offset precision for documents
e, f	Gamma priors
alpha (α)	Dirichlet prior
p	Bernoulli parameter for BOPE
iter_infer	Number of inference steps during an estimation of document proportions within BOPE
iter_train	Number of epochs to train the model

Table 8.2: CTMP input parameters and their description.

Note that the only parameters which are entirely dependent on the dataset are *num_movies, num_words*, and *user_size*. Thus, they reflect according to the size of MovieLens 20M and NETFLIX datasets. Other parameters are modifiable hyperparameters.

8.2.2 Running on Google Cloud

After we have prepared all necessary input data and set input model parameters, we will run the CTMP models on Google Cloud's Compute Engine which is a computed service that lets us create and run virtual machines on Google's infrastructure. Google Cloud offers a wide range of compute engine variants with a variety of configuration options. For our model, we require a large amount of RAM (Random access memory) because the dataset is large and we perform 5-fold cross-validation in parallel. Therefore, the most relevant machine type for our work is shown in Table *8.3* below:

Machine family	GENERAL-PURPOSE
Series	N1
Machine type	n1-highmem-16
CPU	16x Intel® optimized with Intel® MKL and CUDA 11.0
RAM memory	104 GB
Boot Disk	Standard Persistent Disk 100GB

Table 8.3: Specifications of Google Cloud Virtual Machine for training CTMP.

Generally, the time per iteration of the model depends on which dataset and parameters have been selected and the effectiveness of the code written in Python. We have optimized our code by utilizing many techniques to reduce the time complexity and increase the execution speed of the CTMP model. Trainig time for both datasets is shown in Table *8.4* below:

MovieLens 20M	~ 4.2 hours
NETFLIX	~ 6.9 hours

Table 8.4: CTMP training time for datasets.

8.3 Model Evaluation

After running and fitting the model on both datasets with various parameters each time, we now continue with evaluating the performance of these CTMP models. Performance evaluation will be divided into the following parts:

1. Interpretability of learned topics
2. Evaluation metrics and predictive performance
3. Sparsity in topic proportion estimates
4. Sensitivity to hyperparameters

8.3.1 Interpretability of topics

As mentioned before, by incorporating LDA into the CTMP model, interpretable topics can be obtained. This can be seen in Table *8.5* and Table *8.6* below:

MovieLens 20M	
Topic 1	man, father, young, wife, mother, daughter, son, brother, new, old
Topic 2	new, life, one, love, finds, girl, school, two, job, friend
Topic 3	get, new, johnny, one, house, wife, town, police, killer, back

Topic 4	one, two, begins, friend, girl, mother, man, young, life, friends
Topic 5	new, life, world, story, young, love, find, two, help, husband
Topic 6	life, one, new, father, get, woman, friend, two, mother, family
Topic 7	film, documentary, world, story, life, one, new, interviews, footage, history
Topic 8	life, family, town, one, love, home, old, new, war, man
Topic 9	young, family, one, husband, police, mother, father, wife, finds, woman
Topic 10	town, love, new, family, young, father, man, old, two, find

Table 8.5: Top-10 words of first 10 topics learnt by CTMP on MovieLens 20M with parameters of {K=100, lamb=1, alpha=1, p=0.9}.

NETFLIX	
Topic 1	wife life one team man goes story esmeralda get old
Topic 2	year town takes life friends family michael one must son
Topic 3	life father mother new young family scott son finds friend
Topic 4	find killer new town victims years become one back help
Topic 5	new one time children island get years girl find long
Topic 6	man new one bill father york wife son two love
Topic 7	life love two time get one new find jonathan george
Topic 8	new agent drug back fbi cia help american two friends
Topic 9	find new back one love also two family life get
Topic 10	life man take wants new young back money help school

Table 8.6: Top-10 words of first 10 topics learnt by CTMP on NETFLIX with parameters of {K=100, lamb=1, alpha=1, p=0.9}.

8.3.2 Evaluation metrics and predictive performance

The model's predictive performance is measured by its ability to recommend in-items and cold-items. Note that in-items are those containing information from user ratings while cold-items do not possess such information. Thus, recommending items that are all in-items is called **in-matrix prediction,** whereas recommending both in-items and cold-items is called **out-of-matrix prediction**. Both prediction types are evaluated by recall and precision for all users in the test set, and they are measured from top-M recommendations. The top-M recommendation includes items with predicted ratings that are among the M highest. We will denote precision- and recall-at-M with **prec@M** and **recall@M**, respectively:

- $recall@M = \frac{1}{U}\sum_u \frac{M_u^c}{M_u}$ where M_u^c is the number of correct items that appear in Top-M recommendation for user u, M_u is the number items that user u had rated positive.
- $precision@M = \frac{1}{U}\sum_u \frac{M_u^c}{M}$ where M_u^c is the number of correct items that appear in Top-M recommendation for user u, M is the Top-M number.
-

We continue with computing the recall and precision with different combinations of hyperparameters. A domain for each hyperparameter is shown in Table *8.7* below:

K	∈{50, 100}
tops	∈{10}
lamb (λ)	∈ {1, 100}
e, f	∈{0.3}
alpha (α)	∈{1, 0.01}
p	∈{0.7, 0.9}
iter_infer	∈ {50, 100}
iter_train	∈ {25, 50, 100}

Table 8.7: CTMP hyperparameters and their domain.

Note that we set Bernoulli parameter $p \in (0.7, 0.9)$ and did not include 0.5 since this parameter replaces the uniform distribution of OPE on likelihood and prior and adjusting it to 0.5 would result in a BOPE algorithm approximating OPE which is not our goal.

Figure *8.7* andTable *8.6* Figure *8.8* show the recall and precision graphs for in-matrix and out-of-matrix predictions on MovieLens 20M dataset when the model is run with different hyperparameters.
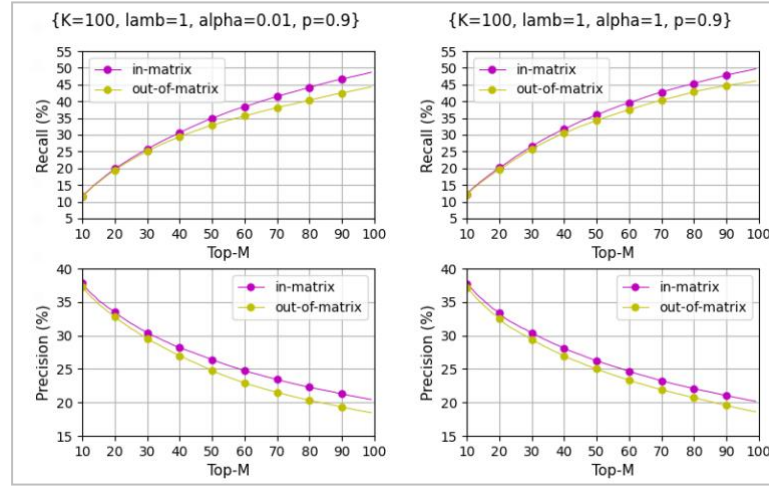
Figure 8.7: Recall & precision graphs of CTMP model on MovieLens 20M dataset. Parameters *K, lamb,* and *p* are fixed. Dirichlet prior *alpha* is varying as {0.01, 1}.
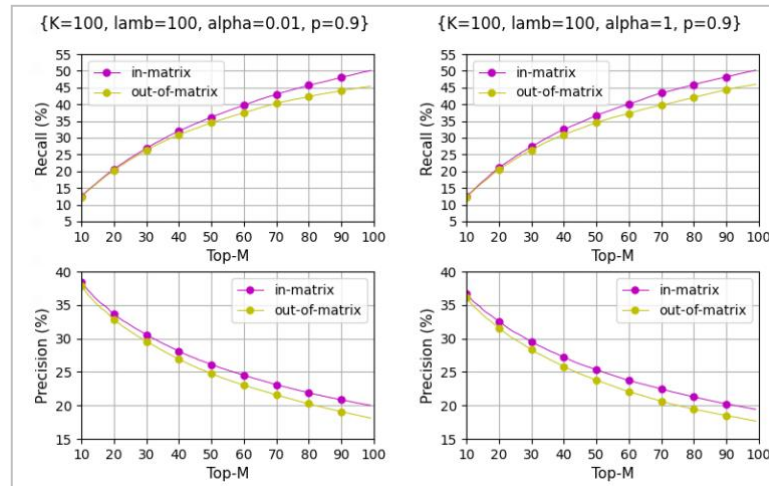
Figure 8.8: Recall & precision graphs of CTMP model on MovieLens 20M dataset. Parameters *K, lamb,* and *p* are fixed. Dirichlet prior *alpha* is varying as {0.01, 1}.

We observe that, the model performance for MovieLens 20M dataset is quite stable with different hyperparameters. The graphs differ from each other with $1 - 3\%$.

The recall graphs reach up to ~50% and ~45% of for in-matrix and out-of-matrix predictions on MovieLens 20M dataset, respectively. These recall performances are approximately same as the recall performance of the CTMP model presented in original paper (refer to [1]). **However, our CTMP model performs better in terms of the precision for top-M recommendations on MovieLens 20M dataset.** In comparison with the original paper's precision graph, our graph shows on average ~7-8% higher precision for each top-M recommendation.

Furthermore, Figure *8.9* below shows the results with parameter *K* set to 50 and varying Bernoulli parameter *p*. With this hyperparameter configuration, we still get approximately the same results as before, except this time there is a slight increase in precision for out-of-matrix predictions. Furthermore, in-matrix and out-of-matrix predictions are closer to each other. The reason behind this may be the parameter *K* being 50 instead of 100.
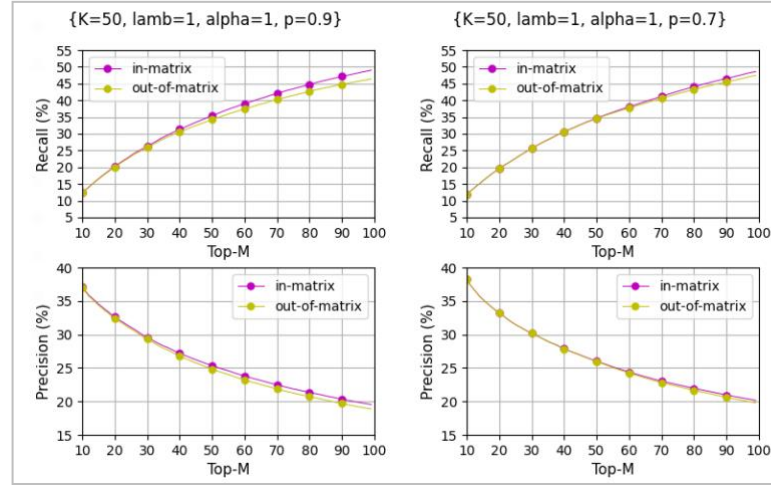
Figure 8.9: Recall & precision graphs of CTMP model on MovieLens20M dataset. Parameters *K, lamb,* and *alpha* are fixed. Bernoulli *p* is varying as {0.7, 0.9}.

Next, we continue fitting the CTMP model on NETFLIX dataset. Figure *8.10*, Figure *8.11* and Figure *8.12* below illustrate the recall and precision graphs with different hyperparameter configurations. Again, from the graphs, we see that the CTMP model performs stable under different hyperparameters. Also, the reason why in-matrix and out-of-matrix lines overlap on the graphs is that the number of cold items for NETFLIX dataset is very small. This makes in-matrix and out-of-matrix predictions similar to each other.
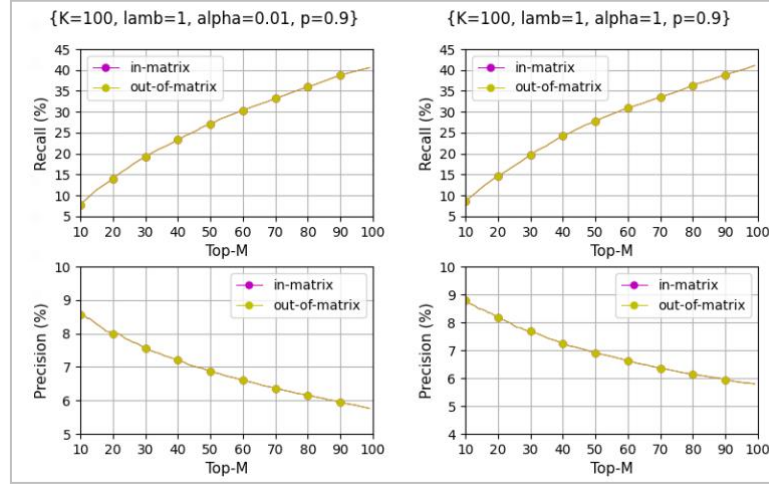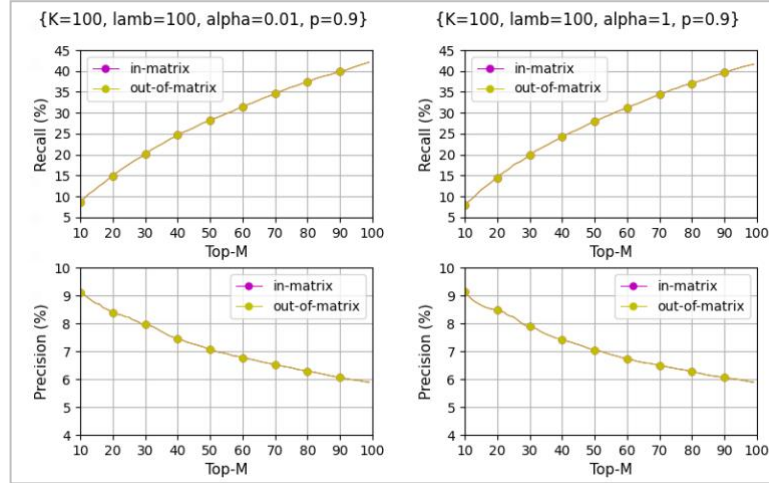
Figure 8.10: Recall & precision graphs of CTMP model on NETFLIX dataset. Parameters *K, lamb,* and *p* are fixed. Dirichlet prior alpha is varying as {0.01, 1}.
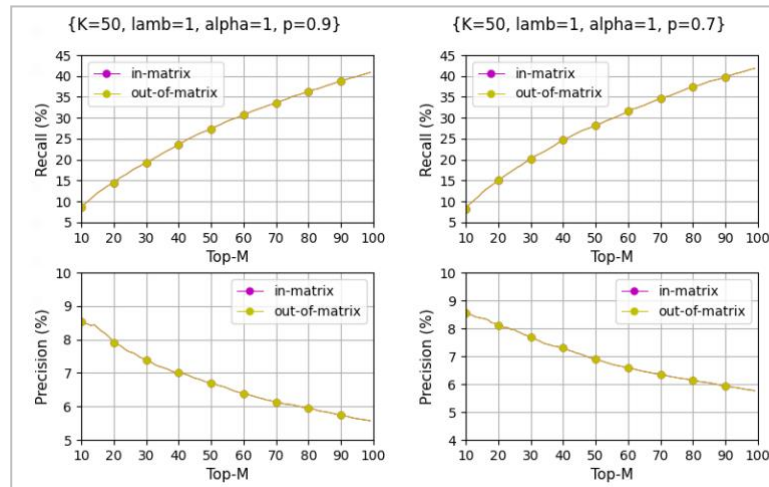

Figure 8.11: Recall & precision graphs of CTMP model on NETFLIX dataset. Parameters *K, lamb,* and *p* are fixed. Dirichlet prior *alpha* is varying as {0.01, 1}.

Figure 8.12: Recall & precision graphs of CTMP model on NETFLIX dataset. Parameters *K, lamb*, and *alpha* are fixed. Bernoulli *p* is varying as {0.7, 0.9}.

8.3.3 Sparsity

After learning the topic proportions, we discovered that a substantial number of θ_j dimensions have *near-zero* values. Near-zero is defined as any value which is less than 10^{-20}. Furthermore, any estimates with near-zero dimensions are considered "sparse" estimates. Figure *8.13* and Figure *8.14* illustrate the sparsity of the learned topic proportions for MovieLens 20M and NETFLIX datasets, respectively.
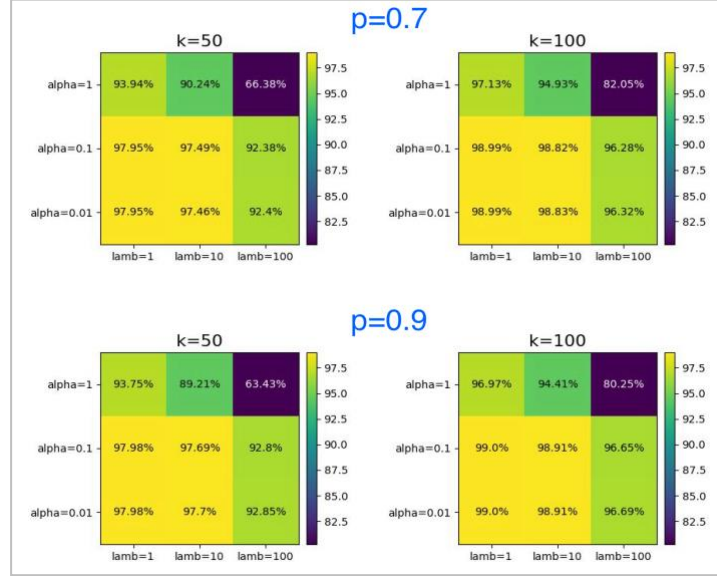
Figure 8.13: Sparsity of learned topic proportions for MovieLens 20M dataset.
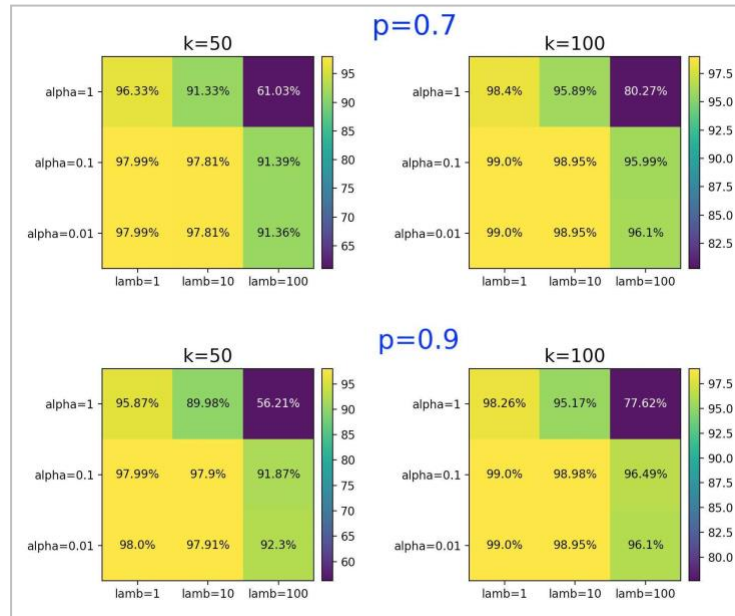
Figure 8.14: Sparsity of learned topic proportions for NETFLIX dataset.

We see that the CTMP model produces sparse estimates of topic mixtures for both datasets. Note that the Dirichlet prior alpha (α) helps to control the sparsity of the topic proportions for each item. The smaller it is, the sparser the topic proportions are.

8.3.4 Sensitivity to hyperparameters

Throughout the experimental studies for recall and precision, we see that the CTMP model is stable and robust with different hyperparameters. In detail, the impact of the parameter K is slightly more obvious than of other parameters.

Furthermore, it is important to note that our CTMP model started to fit the datasets after around 25th iteration instead of 50th-100th iteration which was the expected interval of convergence in the original CTMP paper. The reason behind is that using BOPE instead of OPE for the MAP estimation makes the model convergence much faster and at the same time, prevents the overfitting.

8.4 Transfer Learning

We select the movie - The Naked City (1948). Its plot is as follows:

"Amid a semi-documentary portrait of New York and its people, Jean Dexter, an attractive blonde model, is murdered in her apartment. Homicide detectives Dan Muldoon and Jimmy Halloran investigate. Suspicion falls on various shifty characters whom all prove to have some connection with a string of apartment burglaries. Then a burglar is found dead who once had an elusive partner named Willie. The climax is a very rapid manhunt sequence. Filmed entirely on location in New York City."

Note that, on both MovieLens 20M and NETFLIX datasets, this movie is included. We begin by separately training a CTMP model on both datasets. Modifiable model parameters of settings files are shown in Table *8.8* below:

CTMP for MovieLens 20M	CTMP for NETFLIX
num_topics: 100	num_topics: 50
tops: 10	tops: 10
lamb: 1	lamb: 1
e: 0.3	e: 0.3
f: 0.3	f: 0.3
alpha: 1	alpha: 1
bernoulli_p: 0.9	bernoulli_p: 0.9
iter_infer: 100	iter_infer: 100
iter_train: 50	iter_train: 50

Table 8.8: CTMP hyperparameter selection for transfer learning.

After 50 iterations of training, both models are fit to the respective dataset. Let's look at the top10 words of the topics learned from MovieLens 20M and NETFLIX datasets.

Following that, we compare the topic proportions estimated by both models for "The Naked City (1948)" movie. Estimated topic proportions are shown in Figure *8.15* below:
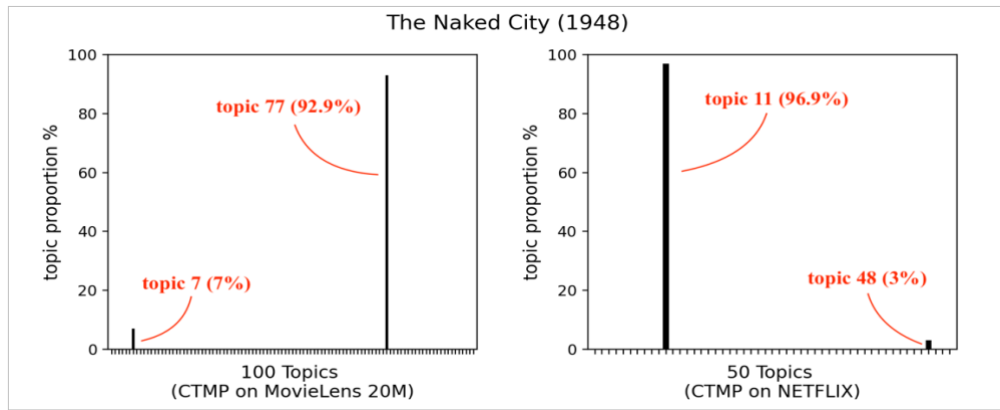
Figure 8.15: Transfer learning between MovieLens 20M and NETFLIX datasets for "The Naked City (1948)" movie.

As illustrated in the figure above, the CTMP model trained on MovieLens 20M estimates that "The Naked City (1948)" movie has 92.9% content related to topic 77 and 7% of content related to topic 7 (the remaining 0.1% comes from other topics, which we disregard because it is insignificant). Now, let's look at what those significant topics are about by looking at the top-10 words of each topic learned by the model:

- **Topic 77 (92.9%)** – killer, police, detective, serial, one, murder, murders, case, young, two.
- **Topic 7 (7%)** – film, documentary, world, one, life, new, interviews, story, history, footage.

We can see that model estimates that the movie will be mostly centered on the keywords murder, detective, police, killer, and so on. Moreover, it also provides little topic proportion about the documentary, cinema, footage, etc. When these results are compared to the movie's original plot, we see that the model accurately estimates the topic proportions of the movie. Topic 7 being assigned a 7% share is also a good sign, because related sentences to this topic were only mentioned in the first and last sentences of the original plot, and they were not particularly relevant to the broader narrative

On the other hand, a CTMP model trained on the NETFLIX dataset suggests that "The Naked City (1948)" movie contains 96.9% content associated to topic 11 and 3% content belonging to topic 48 (remaining 0.1 percent comes from other topics, which we disregard because it is insignificant). Let's look at what those important topics are about by checking the top-10 words of each topic learned by the model:

- **Topic 11 (96.9%)** – police, murder, killer, detective, new, man, life, one, crime, father.
- **Topic 48 (3%)** – jimmy, new, one, film, life, father, alison, time, george, make.

It is apparent that this model, too, accurately approximates the movie's content by giving the most weight to Topic 11, which is concentrated on the phrases police, murder, killer, and so on. 3% being assigned to Topic 48 is again a good sign that this model, too, could associate another negligible topic regarding the first and last sentences of the original plot.

Finally, when we compare the outputs of the two models, we can see that they both learned the topic proportions **accurately and similarly**. They both gave extremely high proportion to the topic, which covers most of the movie's content, and very little weight to the topic, which is not the primary motivation for the movie's plot.

9 Conclusion

In this thesis, we discussed various probabilistic models for recommender systems where we implemented CTMP model augmented with BOPE algorithm. Based on our empirical studies in section 8.3, we see that augmentation of BOPE algorithm to the CTMP model is successful. Despite the plots of movies which are used for content representation were mostly short text, our CTMP model did not overfit the data. The model is stable, fast, and capable of transfer learning. It produced interpretable topics and high recall and precision scores. It also provided sparse content representation which is essential in industrial settings for near-real time recommendation.

Bibliography

[1]   H.M. Le, S.T. ong, Q.P. The, N.V. Linh and K. Than. Collaborative Topic Model for Poisson distributed ratings. *International Journal of Approximate Reasoning,* vol. 95, pp. 62-76, 2018.

[2]   R. Burke and A. Felfernig. Constraint-based recommender systems: Technologies and research issues. *Proceedings of the 10th international conference on Electronic commerce*, pp. 1-10, 2008.

[3]   A. Felfernig, M. Jeran, G. Ninaus and F. Reinfrank. Toward the Next Generation of Recommender Systems: Applications and Research Challenges. 2013.

[4]   Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, vol. 42, pp. 30-37, 2009.

[5]   D. Blei, A. Ng and M. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, vol. 3, pp. 601-608, 2001.

[6]   R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. *Proceedings of the 25th International Conference on Machine Learning,* vol. 25, pp. 880-887, 2008.

[7]   B. Bayar, N. Bouaynaya and R. Shterenberg. Probabilistic non-negative matrix factorization: Theory and application to microarray data analysis. *Journal of bioinformatics and computational biology*, vol. 12, 2014.

[8]   R. Salakhutdinov and A. Mnih. Probabilistic Matrix Factorization. *Proceedings of the 20th International Conference on Neural Information Processing Systems,* pp. 1257–1264, 2007.

[9]   D. Agarwa and B. Chen. fLDA: Matrix factorization through latent dirichlet allocation. *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining,* pp 91-100, 2010.

[10]   C. Wang and D. Blei. Collaborative topic modeling for recommending scientific articles. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 448-456, 2011.

[11]   C. Wang and D. Blei. Variational Inference in Nonconjugate Models. *Journal of Machine Learning Research*, vol. 14, 2012.

[12]   P. Gopalan, L. Charlin and D. Blei. Content-based recommendations with Poisson factorization. *Advances in Neural Information Processing Systems,* vol 4, pp. 3176-3184, 2014.

[13]   P. Gopalan, J. Hofman and D. Blei. Scalable Recommendation with Poisson Factorization. 2013.

[14]   Topic model. https://en.wikipedia.org/wiki/Topic_model. Accessed: 2022-05-27.

[15]   T. Tran, D. Phung and S. Venkatesh. Preference Networks: Probabilistic Models for Recommendation Systems. 2014.

[16]   S. Tu. The Dirichlet-Multinomial and Dirichlet-Categorical models for Bayesian inference. https://stephentu.github.io/writeups/dirichlet-conjugate-prior.pdf. Accessed: 2022-05-27.

[17]   David Blei. Variational Inference. https://rllabmcgill.github.io/COMP-652/lectures/lecture-17.pdf. Accessed: 2022-05-27.

[18] J.M. Joyce. Kullback-Leibler Divergence. *International Encyclopedia of Statistical Science*, 2011.

[19] C. Choy. Expectation Maximization and Variational Inference (Part 1). https://chrischoy.github.io/research/Expectation-Maximization-and-Variational-Inference/. Accessed: 2022-05-27.

[20] L. Lee and C. Wang, "Probabilistic Graphical Models", *Spring*, pp. 10-708, 2017.

[21] D. Blei. Variational Inference: Foundations and Innovations. https://www.eurandom.tue.nl/wp-content/uploads/2019/05/Blei_lectures.pdf. Accessed: 2022-05-27.

[22] M. Wainwright and M. Jordan. Graphical Models, Exponential Families, and Variational Inference. Foundations and Trends in Machine Learning. 2008.

[23] K. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. 2008.

[24] Z. Allen-Zhu. Natasha 2: Faster Non-Convex Optimization Than SGD. 2017.

[25] J. Mairal. Stochastic Majorization-Minimization Algorithms for Large-Scale Optimization. 2013.

[26] A. Yuille and A. Rangarajan. The Concave-Convex Procedure. Neural Computation. 2003.

[27] K. Than, T. Doan. Guaranteed inference in topic models. https://arxiv.org/pdf/1512.03308. Accessed: 2022-05-27.

[28] X. Bui, H. Vu, O. Nguyen and K. Than. MAP Estimation With Bernoulli Randomness, and Its Application to Text Analysis and Recommender Systems. *IEEE Access*, vol. 8, pp. 127818-127833, 2020.

[29] D. Mimno, M. Hoffman, D. Blei. Sparse Stochastic Inference for Latent Dirichlet allocation. *Proceedings of the 29th International Conference on Machine Learning,* 2012.

[30] U. Simsekli, R. Badeau, G. Richard and A. Cemgil. Stochastic Quasi-Newton Langevin Monte Carlo. 2016.

[31] M. Jordan, Z. Ghahramani, T. Jaakkola and L. Saul. An Introduction to Variational Methods for Graphical Models. 1999.

[32] N. Johnson, A. Kemp and S. Kotz. Univariate Discrete Distributions. *John Wiley & Sons*, vol. 444, 2005.

[33] K. Than and T. Ho. Fully Sparse Topic Models. 2012.

[34] MovieLens 20M Dataset. https://grouplens.org/datasets/movielens/20m/. Accessed: 2022-05-27.

[35] Netflix Prize. https://en.wikipedia.org/wiki/Netflix_Prize. Accessed: 2022-05-27.

[36] C. May, R. Cotterell, B. V. Durme. An Analysis of Lemmatization on Topic Models of Morphologically Rich Language. https://arxiv.org/pdf/1608.03995. Accessed: 2022-05-27.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

RS   Recommender Systems
CF   Collaborative Filtering
CB   Content Based
LDA   Latent Dirichlet Allocation
fLDA   Matrix Factorization through Latent Dirichlet Allocation
CTR   Collaborative Topic Regression
CTPF   Collaborative Topic Poisson Factorization
CTMP   Collaborative Topic Model for Poisson distributed ratings
EM   Expectation–Maximization
VI   Variational Inference
VBM   Variational Bayesian Methods
MFVI   Mean-field Variational Inference
KL   Kullback-Leibler
ELBO   Evidence lower bound
MAP   Maximum a Posteriori
OPE   Online Maximum a Posteriori Estimation
BOPE   Bernoulli randomness is Online Maximum a Posteriori Estimation
CGS   Collapsed Gibbs Sampling
HAMCMC   Hessian Approximated Markov Chain Monte Carlo
MCMC   Markov chain Monte Carlo
NLP   Natural Language Processing
RAM   Random Access Memory

Appendix

Appendix A. Objective function

$$l = \sum_j^J \log P(\theta_j, w_j \mid \alpha, \beta) + \sum_j^J \log P(\mu_j \mid \theta_j, \lambda)$$
$$+ \sum_u^U \sum_i^J \left(E_{q(y_{uj}, \eta_u)} \log P(r_{uj}, y_{uj}, \eta_u \mid \mu_j, e, f) - E_{q(y_{uj}, \eta_u)} \log q(y_{uj}, \eta_u) \right) \tag{A.1}$$

We will detail each term in l. According to [27], we have:

$$\log P(\theta_j, w_j \mid \alpha, \beta) = (\alpha - 1) \sum_k \log \theta_{jk} + \sum_v c_j^v \log \sum_k \theta_{jk} \beta_{kv} + Constant \tag{A.2}$$

Because $\mu_j \sim \mathcal{N}(\theta_j, \lambda^{-1} I_K)$, then:

$$\log P(\mu_j \mid \theta_j, \lambda) = -\frac{\lambda}{2} \|\theta_j - \mu_j\|_2^2 + Constant \tag{A.3}$$

The last term in l is detailed below:

$$E_{q(y_{uj}, \eta_u)} \log P(r_{uj}, y_{uj}, \eta_u \mid \mu_j, e, f) - E_{q(y_{uj}, \eta_u)} \log q(y_{uj}, \eta_u) \tag{A.4}$$
$$= E_{q(\eta_u)} \log P(\eta_u \mid e, f) + E_{q(\eta_u)} \log P(r_{uj} \mid \mu_j, \eta_u) + E_{q(y_{uj}, \eta_u)} \log P(y_{uj} \mid \mu_j, \eta_u, r_{uj})$$
$$- \sum_k E_{q(\eta_{uk})} \log q(\eta_{uk} \mid \text{shp}_{uk}, \text{rte}_{uk}) - E_{q(y_{uj})} \log q(y_{uj} \mid \phi_{uj} r_{uj})$$
$$= \sum_k E_{q(\eta_{u,k})} \log \frac{f^e}{\Gamma(e)} \eta_{uk}^{e-1} \exp(-f\eta_{uk}) + E_{q(\eta_{uk})} \log \frac{(\eta_u^T \mu_j)^{r_{uj}} \exp(-\eta_u^T \mu_j)}{r_{uj}!}$$
$$+ E_{q(y_{uj}, \eta_u)} \log \frac{r_{uj}!}{\Pi_k y_{ujk}!} \frac{\Pi_k (\eta_{uk} \mu_{jk})^{y_{ujk}}}{(\eta_u^T \mu_j)^{r_{uj}}}$$
$$- \sum_k E_{q(\eta_{u,k})} \log \frac{(\text{rte}_{uk})^{\text{shp}_{uk}}}{\Gamma(\text{shp}_{uk})} (\eta_{uk})^{\text{shp}_{uk}-1} \exp(-\text{rte}_{uk} \eta_{uk}) - E_{q(y_{uj})} \log \frac{r_{uj}!}{\Pi_k y_{ujk}!} \Pi_k \phi_{ujk}^{y_{ujk}}$$
$$= \sum_k \left((e-1) E_{q(\eta_{u,k})} \log \eta_{uk} - f E_{q(\eta_{u,k})}[\eta_{uk}] \right) + \left(r_{uj} E_{q(\eta_{uk})} \log(\eta_u^T \mu_j) - E_{q(\eta_{uk})}[\eta_u^T \mu_j] \right)$$
$$+ \left(\sum_k E_{q(y_{uj}, \eta_u)} y_{ujk} \log(\eta_{uk} \mu_{jk}) - \sum_k E_{q(y_{uj})} \log y_{ujk}! - r_{uj} E_{q(\eta_{uk})} \log(\eta_u^T \mu_j) \right)$$
$$- \sum_k \left(\log \frac{(\text{rte}_{uk})^{\text{shp}_{uk}}}{\Gamma(\text{shp}_{uk})} + (\text{shp}_{uk} - 1) E_{q(\eta_{u,k})} \log(\eta_{uk}) - \text{rte}_{uk} E_{q(\eta_{u,k})}[\eta_{uk}] \right)$$
$$- \left(\sum_k E_{q(y_{uj})}[y_{ujk}] \log(\phi_{ujk}) - \sum_k E_{q(y_{uj})} \log y_{ujk}! \right) + Constant$$
$$= \sum_k \left((e-1) E_{q(\eta_{u,k})} \log \eta_{uk} - f E_{q(\eta_{u,k})}[\eta_{uk}] \right) + \left(-E_{q(\eta_{uk})}[\eta_u^T \mu_j] \right)$$
$$+ \left(\sum_k E_{q(y_{uj}, \eta_u)} y_{ujk} \log(\eta_{uk} \mu_{jk}) \right)$$
$$- \sum_k \left(\log \frac{(\text{rte}_{uk})^{\text{shp}_{uk}}}{\Gamma(\text{shp}_{uk})} + (\text{shp}_{uk} - 1) E_{q(\eta_{u,k})} \log(\eta_{uk}) - \text{rte}_{uk} E_{q(\eta_{u,k})}[\eta_{uk}] \right)$$
$$- \left(\sum_k E_{q(y_{uj})}[y_{ujk}] \log(\phi_{ujk}) \right) + Constant$$

$$= \sum_k \left((e-1)(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk})) - f \frac{\text{shp}_{uk}}{\text{rte}_{uk}} \right) - \sum_k \mu_{jk} \frac{\text{shp}_{uk}}{\text{rte}_{uk}}$$

$$+ \sum_k r_{uj}\phi_{ujk}(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk}) + \log(\mu_{jk}))$$

$$- \sum_k (\text{shp}_{uk} \log(\text{rte}_{uk}) - \log(\Gamma(\text{shp}_{uk}) + (\text{shp}_{uk}-1)(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk}))$$

$$- \text{rte}_{uk} \frac{\text{shp}_{uk}}{\text{rte}_{uk}}) - \sum_k r_{uj}\phi_{ujk} \log(\phi_{ujk}) + Constant$$

$$= \sum_k r_{uj}\phi_{ujk} \log(\mu_{jk}) - \sum_k r_{uj}\phi_{ujk} \log(\phi_{ujk}) + \sum_k (\text{rte}_{uk} - f - \mu_{jk}) \frac{\text{shp}_{uk}}{\text{rte}_{uk}}$$

$$+ \sum_k (r_{uj}\phi_{ujk} + e - \text{shp}_{uk})(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk})) - \sum_k \text{shp}_{uk} \log(\text{rte}_{uk})$$

$$+ \sum_k \log(\Gamma(\text{shp}_{uk})) + Constant.$$

After summing up three terms (A.2), (A.3) and (A.4), we obtain:

$$l(\theta, \mu, \phi, \text{shp}, \text{rte}, \beta) = \sum_j^J \left((\alpha - 1) \sum_k^K \log \theta_{jk} + \sum_v c_j^v \log \sum_k^K \theta_{jk}\beta_{kv} \right) \tag{A.5}$$

$$- \sum_j^J \frac{\lambda}{2} \|\theta_j - \mu_j\|_2^2 + \sum_u^U \sum_j^J \sum_k^K r_{uj}\phi_{ujk} \log(\mu_{jk})$$

$$- \sum_u^U \sum_j^J \sum_k^K r_{uj}\phi_{ujk} \log(\phi_{ujk}) + \sum_u^U \sum_k^K \left(\text{rte}_{uk} - f - \sum_j^J \mu_{jk} \right) \frac{\text{shp}_{uk}}{\text{rte}_{uk}}$$

$$+ \sum_u^U \sum_k^K \left(\sum_j^J r_{uj}\phi_{ujk} + e - \text{shp}_{uk} \right)(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk}))$$

$$- \sum_u^U \sum_k^K \text{shp}_{uk} \log(\text{rte}_{uk}) + \sum_u^U \sum_k^K \log(\Gamma(\text{shp}_{uk})) + Constant.$$

Appendix B. Update ϕ_{uj}

We then maximize the lower bound with respect to ϕ_{uj}:

$$l(\phi_{uj}) = \sum_k r_{uj}\phi_{ujk} \log(\mu_{jk}) - \sum_k r_{uj}\phi_{ujk} \log(\phi_{ujk}) + r_{uj}\phi_{ujk}(\Psi(\text{shp}_{uk}) - \log(\text{rtee}_{uk})) \tag{B.1}$$

with the constraint $\sum_k^K \phi_{ujk} = 1$.

By adding the Lagrange multipliers λ, we obtain:

$$l(\phi_{uj}) = \sum_k r_{uj}\phi_{ujk} \log(\mu_{jk}) - \sum_k r_{uj}\phi_{ujk} \log(\phi_{ujk}) \tag{B.2}$$

$$+ r_{uj}\phi_{ujk}(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk})) + \lambda \left(1 - \sum_{k=1}^K \phi_{ujk} \right)$$

We take the derivative $l(\phi_{uj})$ with respect to $\phi_{ujk}(k \in \{1, \dots, K\})$, and set to zero, then we get:

$$\phi_{ujk} \propto \exp\left\{\log \mu_{jk} + \psi(\text{shp}_{uk}) - \log(\text{rte}_{uk})\right\} \tag{B.3}$$

Appendix C. Update $\text{shp}_{u,k}$ and rte_{uk}

We then maximize the lower bound with respect to $\text{shp}_{u,k}$ and $\text{rte}_{u,k}$:

$$l(\text{shp}_{u,k}, \text{rte}_{u,k}) = \left(\text{rte}_{uk} - f - \sum_j \mu_{jk}\right)\frac{\text{shp}_{uk}}{\text{rte}_{uk}} \tag{C.1}$$

$$+ \left(\sum_j r_{uj}\phi_{ujk} + e - \text{shp}_{uk}\right)(\Psi(\text{shp}_{uk}) - \log(\text{rte}_{uk})) - \text{shp}_{uk}\,\log(\text{rte}_{uk}) + \log\left(\Gamma(\text{shp}_{uk})\right)$$

Now, we take the derivative $l(\text{shp}_{u,k}, \text{rte}_{u,k})$ with respect to $\text{shp}_{u,k}, \text{rte}_{u,k}(k \in \{1, \dots, K\})$, and set to zero. Note that $\frac{\partial l(\text{shp}_{u,k},\text{rte}_{u,k})}{\partial \text{shp}_{u,k}} = \Psi(\text{shp}_{uk})$, we have:

$$\begin{cases} \dfrac{\partial l(\text{shp}_{u,k}, \text{rte}_{u,k})}{\partial \text{shp}_{u,k}} = \left(\text{rte}_{uk} - f - \displaystyle\sum_j \mu_{jk}\right)\dfrac{1}{\text{rte}_{uk}} + \left(\displaystyle\sum_j r_{uj}\phi_{ujk} + e - \text{shp}_{uk}\right)\Psi'(\text{shp}_{uk}) \\[2mm] = 0 \\[2mm] \dfrac{\partial l(\text{shp}_{u,k}, \text{rte}_{u,k})}{\partial \text{rte}_{u,k}} = -\left(\text{rte}_{uk} - f - \displaystyle\sum_j \mu_{jk}\right)\dfrac{\text{shp}_{uk}}{\text{rte}_{uk}^2} + \left(\displaystyle\sum_j r_{uj}\phi_{ujk} + e - \text{shp}_{uk}\right)\dfrac{-1}{\text{rte}_{uk}} \\[2mm] = 0 \end{cases} \tag{C.2}$$

Now the solution is:

$$\begin{cases} \text{shp}_{uk} = e + \displaystyle\sum_j r_{uj}\phi_{ujk} \\[2mm] \text{rte}_{uk} = f + \displaystyle\sum_j \mu_{jk} \end{cases} \tag{C.3}$$