LDA-LFM: A Joint Exploitation of Review Text and Ratings in Recommender Systems

Tatev Karen Aslanyan
Erasmus University Rotterdam
Rotterdam, the Netherlands
tatevkaren@gmail.com

Flavius Frasincar
Erasmus University Rotterdam
Rotterdam, the Netherlands
frasincar@ese.eur.nl

ABSTRACT

Most of the existing recommender systems are based only on the rating data, and they ignore other sources of information that might increase the quality of recommendations, such as textual reviews, or user and item characteristics. Moreover, the majority of those systems are applicable only on small datasets (with thousands of observations) and are unable to handle large datasets (with millions of observations). We propose a recommender algorithm that combines a rating modeling technique (i.e., Latent Factor Model) with a topic modeling method based on textual reviews (i.e., Latent Dirichlet Allocation), and we extend the algorithm such that it allows adding extra user- and item-specific information to the system. We evaluate the performance of the algorithm using Amazon.com datasets with different sizes, corresponding to 23 product categories. After comparing the built model to four other models, we found that combining textual reviews with ratings leads to better recommendations. Moreover, we found that adding extra user and item features to the model increases its prediction accuracy, which is especially true for medium and large datasets.

CCS Concepts

•**Information systems** → **Recommender systems;** •**Personalization** → Retrieval tasks and goals;

Keywords

e-commerce, latent dirichlet allocation, latent factor model, recommender systems, textual reviews

1. INTRODUCTION

Throughout the last decade, the importance of the Web as a medium for business and electronic transactions has increased drastically, forcing IT to rapidly develop as well, making humans' daily life much easier and more efficient. On its turn, this large development in IT has increased the popularity of online shopping and services. Making purchases online instead of buying products from physical shops, which can be very time-consuming, is one of the major consequences of IT development. However, this large increase in online sales has not only led to an increase in the number of customers but also an increase in the number of products and variety of these products. Therefore, when making purchase decisions, users are forced to process large amounts of information. According to [11, 18, 22, 23] this information overload has a big impact on the human decision process and quality. Hence, it affects people's online purchase experience significantly. Therefore, to overcome this problem, one usually relies on suggestions from others, who have more experience on the topic [32]. This idea is used in the recommender systems aiming to employ various sources of information to recommend products to the users by inferring their interests. Besides solving the problem of information overload, the use of recommender systems also results in increased sales, customer satisfaction and loyalty [31], which explains the increasing popularity of these systems. On the one hand, the information overload motivates the use of recommender systems to make the users' online purchases more convenient. On the other hand, the increasing variety of ways that users can discover, evaluate, and review online products motivates companies and researchers to create even more revealing recommender algorithms, which will enable them to sell more products.

The Web enables users to provide their feedback about the product that they have purchased in the form of ratings and textual reviews. Assuming that the past interests and preferences are often good identifiers of future choices, the previous interactions between items and users can be used for predicting which items might be interesting for a user in the future. Therefore, to correctly recommend the users their desired products, one should predict how the user will respond to a new product [1]. Recommender systems are usually categorized as: Collaborative Filtering systems based on rating data [30], Content-Based systems based on textual data [21], and Hybrid systems that combine these two types of systems [6]. Most of the existing recommender systems are of the first type (based only on ratings), and they ignore the enormous information incorporated in the users' review texts [37]. Ignoring such an important source of information, that can potentially increase the accuracy of recommendations, seems not optimal. Moreover, adding extra user- and item-specific information, not included in the ratings or textual reviews, to the recommender system, might also increase the quality of its recommendations [5, 40, 12]. Figure 1 presents the percentage of items having less than 10 ratings and more than 30 words in their review

text per product category in the datasets of the largest e-commerce Amazon.com [26]. We observe that for almost all product categories it holds that at least 80% of items have very few ratings (less than 10), while over 40% of items have long textual reviews (with more than 30 words per review). Therefore, textual reviews can be considered as a potential source of information that can be used to complement the scarce ratings to increase the prediction accuracy of the recommender system.

Differently than in our previous work [2], for which this work is an extension, we provide the pseudocode for the proposed algorithm and details about the parameter optimization represented in the form of derivations in the appendix where you can find the first and the second order derivatives of the proposed objective function with respect to the model parameters. In addition we provide the code of this work including details about data collection, transformation and preparation for the training and testing processes which can be accessed in the featured Github repository[1].

2. RELATED WORK

Although, there exist a large amount of literature regarding recommender systems that are based on a single type of data, such as ratings or textual reviews, there have been only few attempts of combining user-item ratings and textual reviews to uncover the latent rating and latent review dimensions [3, 25, 20, 35, 36]. [35] combined the predictions of a Latent Factor Model (LFM) with the predictions of the neighborhood model to generate more accurate recommendations. A similar approach was taken in case of the recommender system of 'Bellkor's Pragmatic Chaos', the Netflix Prize contest winner [17]. This system compares the watching and searching habits of similar users, and then recommends movies that share the characteristics with movies that are highly rated by that user. Since then, LFM became the most popular Collaborative Filtering techniques used for both rating and item recommendations [29].

Latent Factor Models are faster to compute then neighborhood models. [36] have developed an algorithm called Collaborative Topic Modeling, combining CF and probabilistic topic modeling, which recommends scientific papers to an online community of users. Authors found that the proposed recommender system, based on both contents of articles and users' ratings, performs better than the recommender system based on standard Matrix Factorization methods. Among all the Hybrid recommender systems, one of the most known systems combining ratings with textual reviews for making recommendations is the Hidden Factors and Topics (HFT) algorithm proposed in [25]. HFT combines latent rating dimensions (learned by LFM) with latent review topics (learned by topic modeling technique LDA) to make rating predictions. [25] stated that, the HFT algorithm results in highly interpretative textual labels for the hidden rating dimensions helping to 'justify' ratings with review text, and in increased prediction accuracy of the recommender system.Another example of a recommender algorithm that combines ratings with textual reviews has been introduced in [20], called Ratings Meet Reviews (RMR). The proposed method is a probabilistic generative model combining the topic modeling technique LDA with a MF method for ratings. The main difference between HFT and RMR is the way the authors combine the two models. More specifically, HFT uses the MF method to model the ratings, whereas RMR uses a mixture of Gaussian distributions. [20] found that RMR outperforms the standard Matrix Factorization based approach and results in similar prediction accuracy compared to HFT. The TopicMF algorithm introduced by [3] is also an example of a recommender algorithm combining ratings and reviews to make recommendations for the users. TopicMF uses biased MF for modeling the ratings and uses Non-negative Matrix Factorization (NMF) for modeling the latent topics in the textual reviews. The main difference between this algorithm and the earlier mentioned recommender algorithms is that it uses NMF instead of the LDA as the topic modeling approach. The final example related to the model introduced in this study is the Rating-Boosted Latent Topics (RBLT) algorithm introduced by [34]. RBLT used LDA for extracting topics from the reviews like HFT and RMR and it also uses the MF for modeling the ratings like HFT. The main difference between RBLT and HFT is that HFT uses item features in rating prediction and topic-distributions as a regularization for these item features, whereas the RBLT includes the topic-distributions in the rating prediction procedure but not in the regularization term. [34] found that adding textual reviews to the CF system increases its prediction accuracy significantly.

One similarity that is shared by all the previously surveyed papers is that they all propose to use textual reviews as well as ratings to model item features and user preferences in a shared topic space and consequently bring them into an LFM to generate recommendations. Our research will also be focused on utilizing recommender systems with the MF approach by using product ratings as well as textual reviews of customers.There have been also few attempts of building a recommender system that allows adding user- or item-specific characteristics, not present in the rating or review data [5, 12, 40]. [5] and [12] introduced CF recommender systems that also allow adding user- and item-specific features on the top of the ratings. As extra item and user information [12] used the browsing data. The CF system extension in [12] has been done by adding extra rows and columns to the user-item rating matrix. However, all these extended recommenders that allow adding a user or item features to the system, are all based only on ratings. To our knowledge, there is no study of recommender systems combining ratings and textual reviews that also allow adding extra user or item information to the system. Another limitation of the existing literature is that most of the proposed recommender systems are modeled and implemented on a dataset consisting of very few product categories or a small number of observations.

To address the previously identified limitations we propose a recommender algorithm called LDA-LFM, which combines the topic-modeling technique LDA with the rating-modeling method LFM and allows adding extra user- and item-specific features to make recommendations. LDA-LFM is a gener-

[1]https://github.com/TatevKaren/TatevKaren-data-science-portfolio/blob/main/LDA-LFM-New-Recommender/

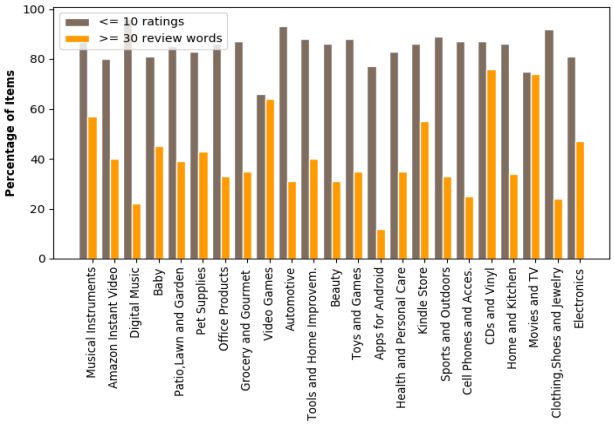

Figure 1: Percentage of ratings and reviews per item. Brown (dark grey for black and white print) bars represent the percentage of items with less than 10 ratings per category. Green (light grey for black and white print) bars represent the percentage of items having on average more than 30 words per product category.

alization of the HFT model proposed by [25], but also uses an alternative approach for model regularization and allows adding extra user- and item-specific features to the recommender system. These extra features will behave as additional factors in the Matrix Factorization driving the ratings following the approach proposed in [12], while these extra features do not appear in the topic modeling method LDA. This system is applicable on both small and large datasets (consisting of millions of reviews), with or without a large number of product categories.

3. RATING AND REVIEW MODELS

In this section, we introduce all models and techniques used to build and evaluate the proposed LDA-LFM model. We describe the technical details and optimization approach of LFM and the topic modeling technique LDA used in this study.

3.1 Latent Factor Model

In Collaborative Filtering recommender systems, Latent Factor Model (LFM), also called Matrix Factorization (MF), have become very popular especially after the earlier mentioned Netflix Prize Contest [14, 15]. Usually, the rating matrix contains lots of missing elements, thus suffers from a sparsity problem. To overcome this problem, LFM uses the idea of dimensionality reduction to estimate and fill in all missing entries of the sparse user-item rating matrix. The goal of dimensionality reduction is to rotate the axis system such that the pairwise correlations between dimensions can be removed and a large sparse matrix can be decomposed into smaller and dense matrices. Accordingly, the reduced, rotated, and complete data matrix representation can be efficiently estimated from a sparse data matrix. The key idea of the Matrix Factorization method is that any m x n sparse matrix R with rank k $<$ min{m,n} can be approximated by rank-k matrices in the following way [33]:

$$R \approx PQ^T \tag{1}$$

where P and Q are m x k and n x k matrices, respectively. So, the user-item sparse matrix R is approximately equal to the product of P and Q matrices, such that the vectors of R can be represented by the rows of matrix P and columns of matrix Q. Stated differently, in LFM, sparse rating matrix R is decomposed into the product of two low-rank rectangular matrices P, the user matrix, and Q, the item matrix, where both P and Q have the same rank k. Each row of matrix P and each column of matrix Q are referred to as *latent factors*. Let us define by p_u the uth row of user matrix P, the user factor representing the affinity of user u towards the rating matrix R, and by q_i the ith row of item matrix Q, the item factor representing the affinity of ith item towards the rating matrix R. Since, some users have a tendency to give higher ratings while other users are more prone to provide lower ratings, and that some products have a tendency to be highly rated compared to other products, baseline predictions (biases) should also be taken into account. [16] referred to biases as the observed variation in rating values due to the effects associated with either items or users independent of any interactions. Correspondingly, estimate of each rating of the uth user about ith item, denoted by r_{ui}, can be expressed as follows:

$$\hat{r}_{ui} = \alpha + b_i + b_u + q_i^T p_u \tag{2}$$

where α represents the global average of all ratings (an offset parameter), b_u and b_i represent the user and item biases, respectively. Accordingly, the error which arises in this estimation is defined as $e_{ui} = r_{ui} - \hat{r}_{ui}$ and in order to learn the latent factors p_u and q_i the following optimization problem should be solved, where we minimize the regularized squared error [17]:

$$\begin{aligned}
&\arg\,\hat{\Theta} \frac{1}{|\mathcal{T}|} \sum_{u,i \in \mathcal{T}} (e_{ui})^2 + \lambda \Omega(\Theta) \\
&\arg\,\hat{\Theta} \frac{1}{|\mathcal{T}|} \sum_{u,i \in \mathcal{T}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \Omega(\Theta) \\
&\arg\,\hat{\Theta} \frac{1}{|\mathcal{T}|} \sum_{u,i \in \mathcal{T}} (r_{ui} - (\alpha + b_i + b_u + q_i^T p_u))^2 + \lambda \Omega(\Theta) \\
&\Omega(\Theta) = \|q_i\|_2^2 + \|p_u\|_2^2 + \|b_i\|_2^2 + \|b_u\|_2^2
\end{aligned} \tag{3}$$

$\mathcal{T}$ represents the corpus of all ratings (in the training set), $\Theta = \{\alpha,\ b_u,\ b_i,\ p_u,\ q_i\}$ is the parameter space of the model. The objective function in Equation 3 can be seen as quadratic loss function which quantifies the loss of accuracy when the element of rating matrix R is approximated by low-rank factorization. $\|.\|_2^2$ represents the squared Frobenius, also called the L_2 norm. We use regularization to prevent model overfitting, which is required especially when the dataset used for fitting the model contains a large number of features, like it is in our case. The constant λ from Equation 3, which is often referred to as regularization constant, determines the level of regularization and controls how hard unnecessary features in the model are penalized. Determining the value of λ is a trade-off between prediction variance and bias. One popular way of determining the optimal value is Grid Search. One of the most popular ways

of solving the optimization problem defined in Equation 3 is Stochastic Gradient Descent (SGD) [16, 25, 38, 39]. Other typical methods which can be considered as possible alternatives to the SGD method, like the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [41] or Orthant-Wise Limited-Memory Quasi-Newton (OWL - QN) [7], work very slowly when the model is fitted on a large training dataset and performing it by one machine is sometimes intractable. SGD addresses these issues because it scales well with both big data and with the size of the model, therefore it is preferred in this analysis. However, even though the method itself is simple and fast, it is known as a "bad optimizer" because it is prone to finding a local optimum instead of a global optimum. A popular technique designed to improve the performance of the SGD method is the Adaptive Moment Estimation (Adam) introduced by [13]. Adam is the extended version of the SGD (with momentum). The main difference compared to the SGD (with momentum), which uses a single learning rate for all parameter updates, is that the Adam algorithm defines different learning rates for different parameters. The algorithm calculates the individual adaptive learning rates for each parameter based on the estimates of the first two moments of the gradients.

3.2 Latent Dirichlet Allocation

Each text review provided by a user, represented as a bag of words, contains valuable information, which can potentially increase the prediction accuracy of the recommender system. For this reason, the textual reviews should be modeled and analyzed. Latent Dirichlet Allocation (LDA) introduced by [4] is one of the most popular text mining methods in the context of recommender systems. Therefore, we will use the LDA as a topic modeling technique in this analysis to uncover the hidden dimensions in the user review texts.There are three main entities defined in this method: words, documents, and corpora. The entity *word* is defined as a basic unit of a discrete data from a vocabulary, $w_{d,j}$ where j = {1,2,...,N_d}, which indicates the index of the word in document d. These words are represented in the form of a vector where the jth element of this vector takes value 1 and the remaining all elements take value 0. The entity *document* is a sequence of N words denoted by $d \in \mathcal{T}$ such that N_d represents the number of words in document d. Finally, the entity *corpus* is defined as a collection of documents denoted by $\mathcal{T}$ = (d_1,d_2,...,d_M), where M is the number of all documents in the corpus.

LDA makes few important assumptions regarding the model. Firstly, it assumes that words carry strong semantic information and that documents discussing similar topics will use similar words. Therefore, latent topics are discovered by identifying a bag of words in a corpus that frequently occur together in a document. Secondly, LDA assumes that documents are probability distributions of latent topics and topics are probability distributions of words. So, every document consists of a certain amount of topics and each of these topics is a distribution of words. Therefore, the model assumes that there are in total K latent topics. Then, LDA assigns to each document d a K-dimensional topic distribution θ_d drawn from a Dirichlet distribution represented in the form of a stochastic vector, such that the kth entry of it, $\theta_{d,k}$, represents the fraction of words in document d which discuss kth topic. Stated differently, the likelihood that words in document d will be about topic k is equal to $\theta_{d,k}$. Furthermore, each topic k is a distribution of words represented by ϕ_k such that each word has a particular likelihood of being used in the topic k. Let us denote by $z_{d,j}$ the topic assigned to the jth word in document d. Then the LDA model is defined as follows:

- $\theta_d \sim \mathcal{DIR}(\gamma)$ with d $\in$ {1,...,M}
- $\phi_k \sim \mathcal{DIR}(\nu)$ with k $\in$ {1,...,K}
- $z_{d,j} \sim$ *Multinomial*(θ_d)
- $w_{d,j} \sim$ *Multinomial*$(\phi_{z_{d,j}})$

where γ represents the parameter of the Dirichlet distribution for document-topic distribution θ_d and ν is the parameter of the Dirichlet distribution for word-topic distribution ϕ_k. $z_{d,j}$ represents the topic assigned to the jth word in document d. We assume that the total number of words in vocabulary is V. Moreover, we denote the likelihood function of $z_{d,j}$ conditional on topic mixture of document d, θ_d, $p(z_{d,j} \mid \theta_d)$ as follows:

$$p(z_{d,j} \mid \theta_d) = \theta_{d,z_{d,j}} \tag{4}$$

Consequently, the probability of jth word in document d, $w_{d,j}$, conditional on the chosen topic z_j denoted by $p(w_{d,j} \mid z_{d,j}, \nu)$ is defined as follows:

$$p(w_{d,j} \mid z_{d,j}, \nu) = \phi_{z_{d,j},w_{d,j}} \tag{5}$$

Furthermore, using the definition of the Dirichlet probability distribution, the conditional topic distribution is defined as follows:

$$p(\theta \mid \gamma) = \frac{\Gamma(\sum_{d=1}^{D} \gamma_d)}{\prod_{d=1}^{D} \Gamma(\gamma_d)} \theta_1^{\gamma_1 - 1} \dots \theta_d^{\gamma_d - 1} \tag{6}$$

where $\theta_d > 0$ and $\Gamma(\cdot)$ represents the Gamma function. Consequently, the joint distribution of a topic θ, K topics z and N words w is defined as follows:

$$p(\theta, z, w \mid \gamma, \nu) = p(\theta \mid \gamma) \prod_{j=1}^{N} p(z_j \mid \theta) p(w_j \mid z_j, \nu) \tag{7}$$

where $N = \sum_{d=1}^{M} N_d$, N_d is the number of words in the document d. Using the properties of discrete and continuous random variables' distributions, the marginal distribution of document d is defined as follows:

$$p(w \mid \gamma, \nu) = \int p(\theta \mid \gamma) \prod_{j=1}^{N} \sum_{z_j} p(z_j \mid \theta_d) p(w_j \mid z_j, \nu) d\theta_d \tag{8}$$

Consequently, using Equations 4, 5 and 8 the likelihood of a text corpus $\mathcal{T}$ conditional on the word distribution ϕ, topic distribution θ_d and topic assignments z is defined as follows:

$$p(\mathcal{T} \mid \gamma, \nu, z) = \prod_{d \in \mathcal{T}} \left(\int p(\theta_d \mid \gamma) \prod_{j=1}^{N_d} \sum_{z_{d,j}} \theta_{d,z_{d,j}} \phi_{z_{d,j},w_{d,j}} \right) d\theta_d \tag{9}$$

This expression can also be rewritten in terms of the topic distribution θ_d and word distribution and ϕ, in the following way:

$$p(\mathcal{T} \mid \theta, \phi, z) = \prod_{d \in \mathcal{T}} \prod_{j=1}^{N_d} \theta_{d,z_{d,j}} \phi_{z_{d,j},w_{d,j}} \tag{10}$$

where parameters θ and ϕ should be estimated, which we denote by Φ, such that $\Phi = \{\theta, \phi\}$. Then, the log-transformation of the conditional corpus probability p($\mathcal{T}$| θ, ϕ, z) is defined as follows:

$$l(\mathcal{T} \mid \theta, \phi, z^*) = \sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} log\big(\theta_{d,z_{d,j}} \phi_{z_{d,j},w_{d,j}}\big) \tag{11}$$

Typically, to estimate the LDA model parameters, Variational Bayesian (VB) methods or sampling approaches based on Markov Chain Monte Carlo (MCMC) sampling are being used [4, 8]. Figure 2 visualizes the dependencies among the LDA model parameters. High γ indicates that it is likely that each document contains a mixture of most of the topics. Conversely, low γ indicates that each document contains only a few of the topics. Furthermore, high ν indicates that each topic contains most of the words of that topic, whereas small ν means that each topic contains only a small amount of words. The parameters γ and ν are at the *corpus level* which are both assumed to be sampled once in the process of corpus generation. The random variable θ_d is the only variable at the *document level*, sampled once per document. Finally, the variables $z_{d,j}$ and $w_{d,j}$ are at the *word level* sampled once for each word per document.

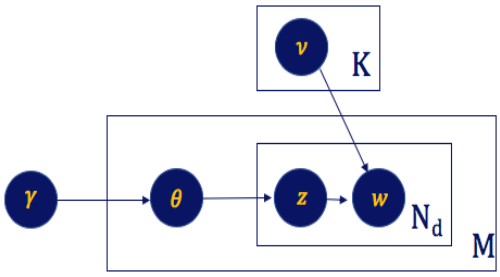

Figure 2: LDA Visualization. K is the number of topics; M is the number of documents; N_d is the length of document d; θ_d represents the topic distribution of document d which follows the Dirichlet distribution with parameter γ; ν is the corresponding parameter of the word probability distribution of the topic k; $w_{d,j}$ is a word in document d at position j and $z_{d,j}$ is that word's topic.

3.3 LDA-LFM Model

The model that we design, called 'Latent Dirichlet Allocation - Latent Factor Model' or shortly LDA-LFM, aims to combine two main core ideas of two methods discussed in Sections 3.1 and 3.2 to uncover both hidden dimensions in ratings and textual reviews, respectively. As it was mentioned earlier, one of the three entities on which topic modeling is based is the *document* entity. Therefore, the concept of 'document' in the LDA-LFM model should be defined properly. There are different ways of defining this concept which should be based on the textual reviews. One can simply consider each text review of user u and item i as a document, denoted by d_{ui}. On the other hand, one can define a document as a set of all reviews corresponding to item i, denoted by d_i. Finally, one can define the document as the set of all reviews provided by a user u as a document, denoted by d_u. [25] found that the second definition, where the concept of a document is defined as the set of all reviews of item i (d_i), leads to the best model performance. The motivation behind this choice is that when users provide feedback about the products in terms of textual reviews, they discuss more often the characteristics of the product rather than discussing their personal preferences. Therefore, we will define the concept of documents in the LDA-LFM in a similar way as in [25].

The idea behind the LDA-LFM model is to find the K-dimensional topic distribution θ_i of each item using textual reviews of item i which shows the extent to which each topic k is discussed across all the reviews for item i. Consequently, these topic distributions are used as item-factors in combination with user-factors in LFM to fully predict all user-item ratings. In Section 3.1 we stated that parameter q_i is the rating factor possessing the properties of item i that can be reviewed by users, whereas in Section 3.2 we stated that parameter θ_i is the topic distribution of words that appear in those reviews. Assuming that, if an item i has a certain property, then it will correspond to a particular topic discussed in that item's textual review, such that $q_{i,k}$ and $\theta_{i,k}$ are positively correlated, we need to define the exact relation between these two parameters. However, $q_{i,k}$ and $\theta_{i,k}$ cannot be considered as being equal since the topic distribution θ_i is a stochastic vector describing topic probabilities while latent item factor q_i can take an arbitrary value in $\mathbf{R}^K$. Stating that q_i is a stochastic vector-like θ_i would result in a loss of power in the proposed model and changing the structure of the topic distribution θ_i to make it more similar to q_i will lead to the loss of probabilistic power in the model. In order to not encounter these problems, the transformation of q_i to θ_i should satisfy monotonicity, $q_i \in \mathbf{R}^K$, and $\sum_k \theta_{i,k} = 1$ assumptions.
The following transformation satisfies all these criteria:

$$\theta_{i,k} = \frac{exp(\kappa q_{i,k})}{\sum_{k'=1}^{K} exp(\kappa q_{i,k'})} \tag{12}$$

where the parameter κ controls for the reaching of the highest possible value of the transformation, often called 'pickiness' parameter. Large value of κ indicates that users discuss only the most important topic, whereas small κ indicates that users discuss all topics equally. We define the transformation, in such a way that, when $\kappa \to \infty$, $\theta_i \to \iota$ (unit vector). Thus, when $\kappa \to 0$, θ_i converges to a uniform distribution. To make sure that the word distribution for topic k (ϕ_k) is a stochastic vector, the following transformation of ϕ_k is defined with an introduction of a new variable ψ:

$$\phi_{k,w} = \frac{exp(\psi_{k,w})}{\sum_{w'} exp(\psi_{k,w'})} \tag{13}$$

where $\psi_k \in \mathrm{R}^V$ is used as a natural parameter for the topic distribution $\phi_k \in \mathrm{R}^V$, where V is the size of the vocabulary.

Correspondingly, it holds that $\sum_n \phi_{k,n} = 1$. Then the objective function of the LDA-LFM model is defined as follows:

$$f(\mathcal{T} \mid \Theta, \Phi, \kappa, z) = \sum_{u,i \in \mathcal{T}} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda(\|p_u\|_2^2 + \|b_i\|_2^2 + \|b_u\|_2^2) - \mu l(\mathcal{T} \mid \theta, \phi, z) \quad (14)$$

where $\Theta = \{\alpha, b_u, b_i, p_u, q_i\}$ and $\Phi = \{\theta, \phi\}$ represent the set of parameters of LFM and LDA model, respectively. The first term of Equation 14 represents the prediction error corresponding to the LFM, the second term represents the regularization of model parameters b_u, b_i, p_u and the third term represents the log-likelihood of the corpus of ratings and users from Equation 11. The parameter $\mu \in R^+$ trades-off the importance of these two effects. We observe that in LDA-LFM model, the regularization of q_i is different compared to the standard Matrix Factorization case, the standard regularization term does not contain the norm of q_i. More specifically, the third term of Equation 14 behaves as a regularization for q_i [25].

3.4 LDA-LFM with Extra Features

As it was mentioned earlier, the proposed recommender system should allow adding extra user- and item-specific features. A key aspect in adding extra features to the system is to better describe users and items, to better predict the preferences of those users for different items. Examples of user features are user demographics such as age, living area, gender, occupation, etc. [9]. If our goal is to build a movie recommender, then the genre, year of its release, name of the director, can all be interpreted as item characteristics, which can be added to the system for making better recommendations. [5], [12] and [40] introduced Collaborative Filtering recommender systems that allow adding user- and item-specific features in addition to ratings. We will follow the approach of [5] and [12], who proposed adding extra rows and columns to the user-item rating matrix representing the extra features added to the LFM. Figure 3

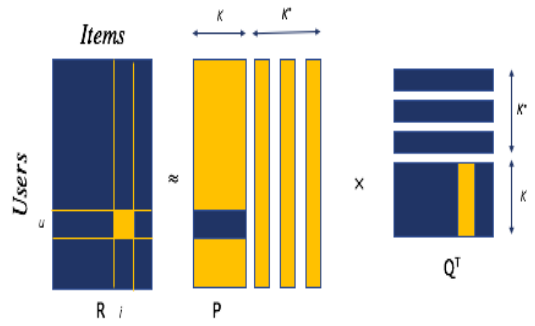

Figure 3: Matrix Factorization of User-Item Rating Matrix with Extra Features. Matrix R is the rating matrix, where the rows of R. Matrices P and Q are the user-factor and item-factor matrices, respectively. To both matrices P and Q are added by K* = 3 extra user-columns and item-rows. These extra K^* columns/rows do not appear in the LDA model while the first K ones do.

visualizes an example of the Matrix Factorization model extended with three extra features. The main idea is to add the same amount of both extra user- and item-specific features. This assumption is necessary because LFM, which is used as a rating modeling technique in the LDA-LFM model, requires matrix multiplication of two matrices with dimensions NUsers x K and K x NItems. This matrix multiplication is only possible when the number of columns in user-factor matrix P is equal to the number of rows of item-factor matrix Q. The extra features denoted by K^* from Figure 3 do not appear in the LDA model and represent non-review factors that affect the review ratings.

3.5 LDA-LFM Model Fitting

Our goal is to find the solution to the optimization problem of Equation 14, which is:

$$\arg_{\Theta,\Phi,\kappa,z} f(\mathcal{T} \mid \Theta, \Phi, \kappa, z) \quad (15)$$

where the corpus $\mathcal{T}$ is given. The LDA-LFM model defines the following iterative stochastic optimization procedure of two steps:

for i in Niter

$$\text{Solve } \arg\min_{\Theta,\Phi,\kappa} f(\mathcal{T} \mid \Theta, \Phi, \kappa, z^{(t-1)})$$
$$\rightarrow \text{Update } \Theta^{(t)}, \Phi^{(t)}, \kappa^{(t)} \quad (16)$$
$$\text{Sample } z_{d,j}^{(t)} \text{ with } p(z_{d,j}^{(t)} = k) = \theta_{d,k} \phi_{k,w_{d,j}}^{(t)}$$

end for

where Niter is the number of iterations, $d \equiv d_{u,i}$ represents the review or set of reviews (document) of item i by user u. In the first step of this optimization procedure from Equation 16 we fix the topic assignments for each word, i.e., the value of latent variable z and we solve the objective function with respect to Θ, Φ and κ. We use the Adam Optimizer for learning the rating related model parameters $\Theta = \{\alpha, b_u, b_i, p_u, q_i\}$, but also the review related parameters $\Phi = \{\theta, \phi\}$, and κ. As it was mentioned earlier, $\theta \in \Phi$ and $q \in \Theta$ are linked through Equation 12. So, we do not use the textual reviews to fit the document-topic distribution θ using the LDA approach. Instead, we determine θ using q, since, we introduced a transformation of ϕ, to ensure that it is a stochastic vector, instead of learning ϕ we learn the parameter ψ. Once we learn ψ, by using the transformation defined in Equation 13, the topic-word distribution ϕ can be determined. Moreover, using the same optimization approach, we also learn the parameter κ.

In the second step of this iterative procedure, using the updated parameter values $\Phi = \{\theta, \phi\}$ determined in the first step by Adam Optimization, we randomly assign a topic k to each word, with a probability that is proportional to the likelihood of the occurrence of that topic with that particular word [36]. That is, the topic assignment probability of assigning kth topic to a word $w_{u,i,j}$ for user u, item i and in jth position $p(z_{w_{u,i,j}} = k)$ is proportional to the product of topic probability for user u, item i ($\theta_{u,i,k}$), and word probability used for that topic ($\phi_{k,w_{u,i,j}}$). We assume that the terms $z_{w_{u,i,j}}$ and $z_{u,i,j}$ are equivalent ($z_{w_{u,i,j}} \equiv z_{u,i,j}$). We iterate through all documents and word positions, d, and j, respectively, to update the corresponding topics assigned to those terms. Finally, we repeat these two steps

for Niter times and report the prediction accuracy of the model corresponding to the last iteration. Following pseudocode describes the fitting process behind the LDA-LFM which includes 2 main steps; initialization and model fitting.

Algorithm 1 LDA-LFM Recommender Algorithm

Input $\leftarrow K, K^*, V, M, N_{iter}, \lambda, \eta, \beta_1, \beta_2, \epsilon, \mu$
rating matrix R, documents, vocabulary
Output: Predicted Rating Matrix
Step 1:

- Determine number of words per document and topic assignment dictionary: itemWords, topics
- Initialize ψ and the parameters in $\Theta = \{\alpha, b_u, b_i, p_u, q_i\}$

Step 2:
for i ***in*** N_{iter} ***do***
for d ***in*** $N_{documents}$ ***do***
for k ***in*** K ***do***
$\theta_{d,k} = \frac{exp(\kappa q_{d,k})}{\sum_{k'} exp(\kappa q_{d,k'})}$
end for
end for
for k ***in*** K $textbf do$
for w ***in*** n_{words} ***do***
$\phi_{k,w} = \frac{exp(\psi_{k,w})}{\sum_{w'} exp(\psi_{k,w'})}$
end for
end for
lk = 0
for d ***in*** M ***do***
for j ***in*** $documentd$ ***do***
$p_z = \theta_{d,.}\phi_{.,j}$
$z_{d,j} \leftarrow$ ***sample from*** $Multinomial(p_z)$
$topics_{[d,j]} = z_{d,j}$
$lk+ = log(\theta_{d,z_{d,j}}\phi_{z_{d,j},j})$
end for
end for
for $user, item, rating$ ***in*** R ***do***
$\hat{r}_{user,item} = \alpha + b_{item} + b_{user} + p_{user,k}q^T_{item,k}$
$e_{user,item} = \hat{r}_{user,item} - rating$
$f = e_{user,item} - \mu lk$
for ω_i ***in*** $b_{user}, b_{item}, p_{user,k}, q_{item,k}, \psi_{k,\omega}$ ***do***
$m_i = \beta_1 m_{i-1} + (1-\beta_1)\frac{\partial f}{\partial \omega_{i-1}}$
$r_i = \beta_2 r_{i-1} + (1-\beta_2)\frac{\partial^2 f}{\partial \omega_{i-1}\partial \omega_{i-1}}$
$\omega_i = \omega_{i-1} - \eta \frac{m_{i-1}}{\sqrt{r_{i-1}+\epsilon}}$
end for
end for
end for
close;

The algorithm requires as an input the number of topics K, the number of extra latent factors for the LFM model K*, the size of the vocabulary V, the number of iterations N_{iter}, the hyperparameter μ, the regularization parameter λ. Moreover, the parameters of the Adam Optimization method (η, β_1, β_2, ϵ) should be initialized too. Finally, the algorithm also requires the rating matrix R, the documents, and the vocabulary, where the vocabulary is only used for initializing the topics assigned to each word in a document.

In Step 1 of the LDA-LFM algorithm, the number of words per document, where each document represents all reviews corresponding to an item, is determined and stored in the *itemWords* vector. The initial topic assignment of all words in the dictionary is stored in *topics*. Moreover, all parameters of the LFM model and ψ of the LDA model are initialized. Finally, we randomly assign a set of topics to each document in the corpus. In Step 2, the LDA-LFM algorithm per iteration generates a document-topic vector θ_i, which determines the topic-distribution, based on the value of item-factors q_i obtained from the MF procedure. Next, the values of $\psi_{k,w}$ are used to generate topic-word probabilities $\phi_{k,w}$. Correspondingly, the algorithm goes through the entire corpus of all documents and randomly assigns topics to all words in that document, using the conditional probability p_z given in Equation 16. Then, the generated document-topic (θ) and topic-term (ϕ) probabilities are used for calculating the corpus likelihood. Finally, all above obtained values are used to predict ratings and update model parameters b_u, b_i, p_u, q_i, ψ, and κ in each iteration. The processes in Step 2 are repeated for N_{iter} times and the prediction accuracy of the last iteration is computed.

4. EVALUATION

As a prediction accuracy measure we use the Mean Squared Error (MSE) determined as follows:

$$MSE = \frac{\sum_{(u,i)\in\mathcal{T}^*}(\hat{r}_{u,i} - r_{u,i})^2}{\mid \mathcal{T}^* \mid} \tag{17}$$

where $\mathcal{T}^*$ represents the corpus of all ratings in the test set, $r_{u,i}$ represents the real rating from the test data for user u and item i, and $\hat{r}_{u,i}$ is corresponding predicted rating. MSE can take only non-negative values. Moreover, a lower value of MSE is an indication of better performing model. It is worth mentioning that the analysis is performed on a commodity machine with a Cori7 processor, 2.2 GHz frequency, and 252Gb memory space using the programming language Python 3.7.

4.1 Data

In this research, we use a collection of datasets provided corresponding to the 23 product categories supplied by one of the largest e-commerce companies in the world, Amazon.com. This data without duplicates was prepared by Julian McAuley. It consists of 142.8 million product reviews and metadata for 9.4 million products, spanning a period of 18 years, from May 1996 to July 2014 [24, 26]. The chosen dataset is of a 5-core type, that is, the data set excludes all customers and products having less than 5 reviews. The review dataset includes feedbacks of Amazon customers in the form of ratings, textual reviews, and helpfulness scores. Meanwhile, the metadata includes various characteristics of the product: price, brand, descriptions, category informa-

tion, image features, and links of 'also viewed', 'also bought' products. The raw review data, after removing duplicates and excluding users or items with less than 5 reviews, consists of 42.13 million reviews. Table 1 presents the general overview of the datasets of all product categories. We observe that all datasets are highly sparse and contain a very large amount of missing ratings. For almost all datasets it holds that the average star rating is approximately equal to 4. Moreover, the average number of words per review is at least 18 and at most 67. Finally, the smallest dataset, *Musical Instruments*, consists of 0.5 million reviews, and the largest dataset, *Electronics*, consists of approximately 8 million reviews.

4.2 Data Preparation

In order to correctly evaluate the chosen model, we split the data into three datasets: training, validation, and test sets. We fit the model on the training data and find a set of optimal model parameters (hyperparmeter tuning) using the validation set. Finally, we use the test set for predicting the ratings and calculating model accuracy measures using the optimal set of parameters from the hyperparmeter tuning. For data separation we use the common 80/20 splitting rule. In order to have enough observations to correctly fit the model, we put 80% of all observations in the training set, while the remaining 20% we equally divided into the test and validation sets. However, splitting the data into training, test and validation sets, when some of the users and items appear only in the test set and not in the training set, will result in a loss of information about those users and items during the training of the model. Therefore, after randomly splitting the data into train and test set, we make sure that there is no user or item that is present in the test set but not present in the training set.

For implementing the topic modeling technique LDA, the review data should be cleaned. Therefore, we perform a few Natural Language Processing (NLP) tasks on the textual reviews in review tuples by using the *Natural Language Tool Kit (NLTK)* library of the programming language Python. Firstly, we apply tokenization to all review texts, which are provided as a group of sentences, and transform them into a group of words. Secondly, we transform them to lower case words and remove from these tokenized reviews the common English stop words and one-letter words. Subsequently, all special characters, digits, punctuation and single or multiple spaces are removed. Next, we apply lemmatization to the processed review text, for removing inflectional endings and holding the dictionary (base) form of a word only, known as the lemma of the word. Finally, we combine all those cleaned reviews corresponding to the same item and create a corpus of documents, where each document contains all reviews (represented in the form of a group of words) corresponding to one item.

4.3 Model Selection

We introduced various parameters in the methodology section which should be initialized. We initialize the offset α by averaging over all ratings in the training set. Vectors b_i, b_u and matrices P, Q are initialized using the random normal distribution. The fitting procedure of all models have been performed by Adam Optimization with the learning rate 0.01. As initial value for κ we take the value 1, which will be updated by Adam Optimization while fitting the model. For each model we run 35 iterations [3] (with 20 iterations), [15] (with 20-35 iterations), [27] (with 30 iterations) while updating model parameters in Θ, Φ, and κ in each iteration. The prediction accuracy of the model is reported based on the last model corresponding to *35*th iteration, assuming that the last model, after all the updates, is the best performing model. As a common practice, for the LDA model we set both parameters γ and ν equal to 0.1. Following the approach of [25] we perform the analysis with number of latent factors in LFM model (K*) and number of topics in the LDA model (L) equal to 5.

LDA-LFM contains two regularization parameters, λ and μ. We tune this set of two parameters using the Grid-Search method, which fits the model for every specified combination of these two parameters and evaluates each of these models using validation set. As a result, the most accurate model specification, per product category, is then used in the main model prediction applied to the test dataset. Following the approach of [25], for λ we use values {0, 0.001, 0.01, 1, 10} as a possible values in the Grid-Search, while for regularization constant μ we use the values {1, 10, 100, 1000, 10,000}. As it was mentioned in Section 3.3, we set the number of documents in the LDA model equal to the number of items in the data, where each document represents all reviews of an item in the training set. We set the size of the vocabulary equal to 5000, by keeping the most 5000 frequent words from the corpus of all documents built from the item reviews present in train set. Table 2 presents the results of the Grid-Search per dataset for the LDA-LFM model with both the number of topics and the number of latent factors being equal to 5 (K = 5). For each dataset, the training set has been used for fitting the model and corresponding validation set has been used for calculating the prediction accuracy of the model for each possible combination of parameters λ and μ. Consequently, per dataset, the pair of parameters λ and μ is chosen, which corresponds to the model with the smallest MSE value. The first column of Table 2 represents the optimal value of regularization parameter λ denoted by λ_{opt}. From the table we can see that for almost all datasets it holds that the optimal regularization parameter is equal to 10. There is only one dataset, *Amazon Instant Video*, for which λ_{opt} is equal to 1. The second column of Table 2 represents the optimal value of hyper-parameter μ denoted by μ_{opt}. We observe that unlike the λ_{opt}, there is no single value of μ which is optimal for the majority of all datasets.

4.4 Baseline Models

In order to test for the performance of the proposed LDA-LFM model, we use 4 other recommender systems based on different algorithms. Then we compare the prediction accuracy of the LDA-LFM model with the performance of the following methods:

***Offset Model*:** the predicted rating for all users is the same and is equal to the global average α.

***Baseline Rating Model*:** the predicted rating $\hat{r}_{ui} = \alpha + \bar{r}_u + \bar{r}_i$ with α representing the global average rating, $\bar{r}_u$

Table 1: Overview of datasets. The following statistics per dataset are reported: number of users (NUsers), number of items (NItems), number of reviews (NReviews), average number of words in textual reviews after removing the stopwords (A. W.), average star rating (A. R.), and sparsity of the user-item rating matrix (Sparsity).

Dataset	NUsers	NItems	NReviews	A. W.	A. R.	Sparsity
Electronics	4,201,696	476,002	7,824,482	43	4.012	0.00039
Clothing, Shoes and Jewelry	3,117,268	1,136,004	5,748,920	26	4.145	0.00016
Movies and TV	2,088,620	200,941	4,607,047	58	4.187	0.00110
Home and Kitchen	2,511,610	410,243	4,253,926	36	4.099	0.00041
CDs and Vinyl	1,578,597	486,360	3,749,004	67	4.403	0.00049
Cell Phones and Accessories	2,261,045	319,678	3,447,249	30	3.811	0.00048
Sports and Outdoors	3,117,268	1,136,004	3,268,695	34	4.145	0.00034
Kindle Store	1,406,890	430,530	3,205,467	42	4.232	0.00050
Health and Personal Care	1,851,132	252,331	2,982,326	33	4.110	0.00063
Apps for Android	1,323,884	61,275	2,638,173	18	3.996	0.00325
Toys and Games	1,342,911	327,698	2,252,771	33	4.150	0.00051
Beauty	1,210,271	249,274	2,023,070	31	4.149	0.00067
Tools and Home improvement	1,212,468	260,659	1,926,047	36	4.130	0.00061
Automotive	851,418	320,112	1,373,768	30	4.185	0.00051
Video Games	826,767	50,210	1,324,753	58	3.979	0.00051
Grocery and Gourmet Food	768,438	166,049	1,297,156	31	4.255	0.00102
Office Products	909,314	130,006	1,243,186	36	3.979	0.00105
Pet Supplies	740,985	103,288	1,235,316	37	4.111	0.00161
Patio, Lawn and Garden	714,791	105,984	993,490	37	4.006	0.00130
Baby	531,890	64,426	915,446	41	4.118	0.00270
Digital Music	478,235	266,414	836,006	41	4.540	0.00066
Amazon Instant Video	426,922	23,965	583,933	28	4.316	0.00571
Musical Instruments	339,231	83,046	500,176	45	4.244	0.00178

the average difference between user ratings and the global average α, $\bar{r}_i$ represents the average difference between item ratings and the global average α.

***LFM*:** standard Latent Factor Model model corresponding to Equation 2.

***LDAFirst*:** in this model the user feedback in the form of ratings will be used as an input for the standard LFM, while the textual reviews will be used as an input for the LDA model described in Section 3.2. The key difference between this method and the proposed method LDA-LFM is that, in LDAFirst the topic-distributions θ_i are sampled from a Dirichlet distribution, where each document is treated as the set of all reviews corresponding to item i, and they are used to set the q_i values, which stay constant while modeling the ratings. Thus we do not learn the q_i parameter of the LFM during the iterative optimization procedure and we only update the parameters b_i, b_u, and p_u using the Adam Optimization. In the LDA-LFM model, we do not use the LDA method for determining the topic-distributions θ_i. After that we sample the word topics, we learn $\Phi = \{\theta, \phi\}$ using Adam Optimization as in Equation 16 (where q_i is dependent on θ_i by means of Equation 12). Since we start this method by the LDA model and use its output (θ_i) as an input for the LFM method (q_i), we refer to this method as LDAFirst.

5. RESULTS

Firstly, we perform the analysis for the case when no extra features are added to the LDA-LFM model, assuming that the number of topics in LDA model is equal to the number of latent factors in LFM with K $\in$ {5,10} [25]. Correspondingly, in order to analyse the impact of adding extra latent features (user and item characteristics) on the performance of recommender system based on the proposed LDA-LFM model, such that the number of topics in the LDA has value 5 and the number of extra latent factors with 4 different values of extra features $K^* \in \{1, 2, 3, 4\}$. Table 3 presents the prediction per product category with the number of topics equal to 5.

We observe that for the majority of supplied datasets it holds that the Offset and Baseline models perform the worst, with large MSE values, compared to the LFM, LDAFirst, and LDA-LFM models. This can also be seen in Figure 4 which visualizes the performance of the LDA-LFM compared to the Offset and BRM, respectively. Only for *Video Games* and *Tools and Home Improvements* datasets the Offset method performs better than the LFM and LDAFirst models. Moreover, we observe that compared to the Offset and Baseline models, standard LFM improves the recommender systems prediction accuracy for almost all datasets, except datasets *Patio, Lawn and Garden*, *Video Games*, and *Tools and Home Improvement*. This can be seen by the large difference between the MSE values corresponding to the LFM, and MSE values corresponding to the Offset and Baseline models. We also observe that the MSE's corresponding to the LFM and LDAFirst models, also visualized in Figure 5, are very close to each other for the majority of datasets. This means that the LDAFirst model does not improve the prediction accuracy a lot compared to the standard LFM model. The LDAFrist slightly outperforms LFM in case of the datasets *Baby,Office Products Grocery and Gourmet Food*, *Apps for Android*, *CDs and Vinyl*. From Table 3 we observe that the LDA-LFM model outperforms all other models in almost all datasets, with its lowest MSE values. Last two columns of Table 3 present the percentage de-

Table 2: Parameter Tuning LDA-LFM. Grid Search parameter tuning for regularization parameters λ and μ per product category dataset. λ_{opt} and μ_{opt} are the optimal parameters chosen from λ = {0, 0.001, 0.01, 1, 10 } and μ = {1, 10, 100, 1000, 10,000} respectively, resulting in the most accurate model for the validation set.

Dataset	λ_{opt}	μ_{opt}
Electronics	10	1000
Clothing, Shoes and Jewelry	10	1000
Movies and TV	10	1000
Home and Kitchen	10	1000
CDs and Vinyl	10	10,000
Cell Phones and Accessories	10	1
Sports and Outdoors	10	1000
Kindle Store	10	1000
Health and Personal Care	10	100
Apps for Android	10	10
Toys and Games	10	1000
Beauty	10	10,000
Tools and Home improvement	10	1000
Automotive	10	10,000
Video Games	10	1000
Grocery and Gourmet Food	10	10
Office Products	10	1000
Pet Supplies	10	10,000
Patio, Lawn and Garden	10	1
Baby	10	1
Digital Music	10	10
Amazon Instant Video	1	100
Musical Instruments	10	100

crease in the MSE of the LDA-LFM model compared to the LFM and the LDAFirst models, respectively. Both improvement columns consist mostly of positive entries. We observe that the proposed LDA-LFM results in at least 0.24% (*Health and Personal Care*) and at most 14.12% (*Kindle Store*) improvement in prediction accuracy, compared to the standard LFM. From Imp.$_{[5]/[4]}$ we observe that the proposed LDA-LFM results in at least 0.28% (*Electronics*) and at most 14.12% (*Kindle Store*) improvement, compared to the LDAFirst model. The improvement columns in Table 3 contain also few negative values which correspond solely to small datasets (*Musical Instruments*, *Amazon Instant Video*, *Patio, Lawn and Garden*). Table 3 also reports the average MSE over all datasets per model in case of K = 10. We observe that average MSE's per model with K = 5 and K = 10 are similar.

Table 4 presents the prediction results in terms of MSE, per product category with the number of topics equal to 5 and extra added features. We observe that for almost all datasets there is at least one LFM-LDA model with extra feature(s) with higher prediction accuracy (lower MSE value compared to the corresponding MSE value in the K* = 0 model, i.e., without extra features). Moreover, for the datasets *Musical Instruments*, *Patio, Lawn and Garden*, *Automotive*, *Toys and Games*, *Health and Personal Care*, *Sports and Outdoors*, *CDs and Vinyls*, *Home and Kitchen* and *Movies and TV* all four models with different number of extra features are performing better compared to the model without extra added features. However, there are few datasets (*Amazon Instant Video*, *Office Products*, and *Beauty*) for which it holds that adding extra features to the LDA-LFM model either does

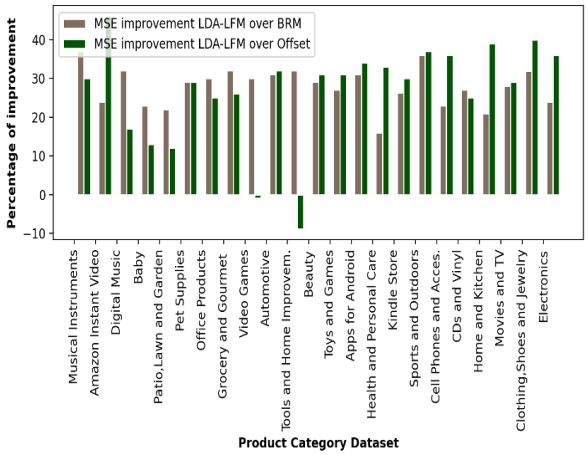

Figure 4: Percentage decrease in MSE when comparing LDA-LFM model performance to the Offset model (green bars) and to the BRM model (grey bars) performances.

not change or worsens the performance of the model. We observe that all those datasets, for which adding extra features is not efficient, are either very small or medium size datasets in the set of all 23 Amazon datasets used in this study. The last row of Table 4 presents the number of cases in which adding a particular amount of extra features leads to an increase in the prediction accuracy of the model. We observe that in all four cases (adding 1, 2, 3, and 4 extra features), the number of datasets with better performance are close to each other. More specifically about 15 out of 23 datasets (from which around 9 cases corresponds to a medium or a large dataset), adding extra features results in more accurate recommendations.

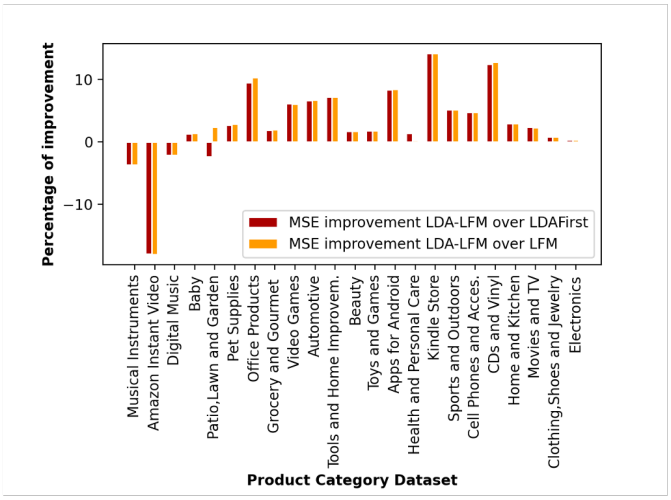

Figure 5: Percentage decrease in MSE when comparing LDA-LFM model performance to the LFM model (orange bars) and to the LDAFrist model (red bars) performances.

Table 3: Prediction results in terms of MSE with K = 5 number topics. Imp.[5]/[3] reports the percentage improvement of LDA-LFM model compared to the LFM model, in terms of prediction accuracy. Imp.[5]/[4] shows the percentage improvement of LDA-LFM model compared to the LDAFirst model. The average MSE per model is also reported for the K = 10 case.

Dataset	Offset[1]	Base.[2]	LFM[3]	LDAFirst[4]	LDA-LFM[5]	Imp. [5]/[3]	Imp.[5]/[4]
Electronics	2.909	2.345	1.789	1.789	1,780	0.28%	0.28%
Clothing, Shoes and Jewel.	2.275	2.122	1.456	1.457	1.445	0.76%	0.82%
Movies and TV	2.803	2.334	1.721	1.723	1.682	2.27%	2.38%
Home and Kitchen	2.535	2.249	1.841	1.841	1.787	2.93%	2.93%
CDs and Vinyl	2.508	2.083	1.746	1.740	1.523	12.77%	12.47%
Cell Phones and Access.	2.542	2.448	1.996	1.995	1.901	4.76%	4.71%
Sports and Outdoors	2.138	2.096	1.422	1.422	1.349	5.13%	5.13%
Kindle Store	2.516	2.145	1.814	1.813	1.581	14.12%	14.12%
Health and Personal Care	2.392	2.259	1.670	1.689	1.666	0.24%	1.36%
Apps for Android	2.984	2.397	2.190	2.188	2.006	8.40%	8.32%
Toys and Games	2.258	2.152	1.512	1.512	1.485	1.79%	1.79%
Beauty	2.371	2.256	1.675	1.674	1.646	1.73%	1.67%
Tools and Home Improve.	1.392	2.136	1.634	1.634	1.516	7.22%	7.22%
Automotive	2.081	2.064	1.511	1.511	1.410	6.68%	6.62%
Video Games	1.603	2.354	1.721	1.722	1.617	6.04%	6.10%
Grocery and Gourmat	2.020	2.129	1.520	1.519	1.491	1.91%	1.84%
Office Products	2.168	2.306	1.813	1.796	1.626	10.31%	9.47%
Pet Supplies	2.480	2.265	1.815	1.814	1.765	2.81%	2.70%
Patio, Lawn and Garden	1.970	2.235	1.780	1.781	1.738	2.36%	-2.41%
Baby	1.934	2.194	1.717	1.715	1.693	1.40%	1.28%
Digital Music	1.261	1.555	1.030	1.030	1.052	-2.14%	-2.14%
Amazon Instant Video	2.828	1.985	1.278	1.277	1.508	-18.09%	-18.00%
Musical Instruments	1.735	1.921	1.165	1.165	1.208	-3.69%	-3.69%
Average MSE K = 5	2.248	2.122	1.643	1.644	1.572		
Average MSE K = 10	2.248	2.122	1.637	1.637	1.579		

Table 4: Prediction results of LDA-LFM model in terms of the MSE with K = 5 number topics. K^* defines the number of extra factors added to the LDA-LFM model. $K^* = 0$ represents the case when no extra feature has been added to the model, results corresponding to the Table 3. $K^* = 1$, $K^* = 2$, $K^* = 3$, and $K^* = 4$ correspond to the LDA-LFM model predictions with 1, 2, 3, and 4 extra features, respectively.

Dataset	$K^*=0$	$K^*=1$	$K^*=2$	$K^*=3$	$K^*=4$
Electronics	1.78019	**1.78001**	**1.77651**	**1.77901**	1.78199
Clothing, Shoes and Jewel.	1.44511	**1.44510**	**1.44515**	**1.44510**	1.44581
Movies and TV	1.68230	**1.68204**	1.69210	1.69199	**1.68221**
Home and Kitchen	1.78696	**1.78694**	**1.78690**	1.78710	**1.78691**
CDs and Vinyl	1.52294	**1.52293**	**1.52278**	**1.52280**	**1.52270**
Cell Phones and Access.	1.90113	1.90553	1.90540	**1.90112**	**1.90154**
Sports and Outdoors	1.34960	**1.34959**	**1.34950**	**1.39512**	**1.39516**
Kindle Store	1.58104	1.58118	1.58143	**1.58100**	**1.58011**
Health and Personal Care	1.66564	**1.66558**	**1.66557**	**1.66555**	**1.66553**
Apps for Android	2.00655	**2.00617**	**2.00524**	**2.00644**	2.00696
Toys and Games	1.48472	**1.48455**	**1.48458**	**1.48442**	**1.48446**
Beauty	1.64578	1.64584	1.64580	1.64578	1.64583
Tools and Home improve.	1.51640	1.51644	**1.51637**	1.51658	1.51656
Automotive	1.40996	**1.40983**	**1.40977**	**1.40973**	**1.40984**
Video Games	1.61691	1.61699	**1.60912**	**1.60902**	**1.60936**
Grocery and Gourmet	1.49049	1.49057	**1.49046**	**1.49045**	1.49084
Office Products	1.62635	1.62666	1.62673	1.62674	1.62668
Pet Supplies	1.76503	**1.76502**	1.76501	1.76500	**1.76515**
Patio, Lawn and Garden	1.73829	**1.73810**	**1.73812**	**1.73811**	**1.73809**
Baby	1.69345	**1.69332**	1.69358	**1.69272**	1.69372
Digital Music	1.05179	**1.05165**	**1.05175**	1.05185	**1.05170**
Amazon Instant Video	1.50755	1.50852	1.50949	1.50835	1.50885
Musical Instruments	1.20791	**1.19803**	**1.19825**	**1.19818**	**1.19817**
Number of Times Beneficial	-	15	15	15	14

6. CONCLUSION

Most of the existing recommender systems are based on the ratings data only, since incorporating textual reviews in a rating based model is not an easy task to perform, while reviews, opinions, and shared experiences of the consumer represent a rich source of information about the consumer preferences. Moreover, most of those recommender systems are tested and applied only on a dataset with a small number of observations (with thousands of observations) and are not applied to a very large dataset (with millions of observations). Finally, the few recommender systems in the existing literature that allow adding extra user- and item-specific features to the recommender system, are mainly CF type of systems based on ratings only. Hence, none of those systems combine ratings with textual reviews. Therefore, taking into account all those limitations in the existing literature, we utilize the Latent Factor Model by using both ratings and textual reviews of customers, such that it is applicable on both small and large datasets and also allows adding user- and item-specific data to it. We have shawn how one can combine review-based LDA for topic modeling, with a rating-based LFM for rating predictions.

From the prediction results where we compared the prediction accuracy of the proposed LDA-LFM model to the prediction accuracy's of various baseline models applied on various datasets.We found that adding textual reviews to the recommender system leads to an increased prediction accuracy, which is especially true for medium and large datasets. Then we introduced an approach of adding extra latent features to the user-item rating matrix of the proposed LDA-LFM model, representing the user- and item-specific features not present in the review data. We found that for the majority of datasets (15 out of 23 datasets) it holds that, adding extra features to the proposed recommender system increases the quality of its recommendations, resulting in lower MSE, thus higher prediction accuracy. This indicates that again, that the improvements are better visible in medium and large datasets. In order to add extra user and item characteristics to the proposed LDA-LFM model, we used the approach of adding extra rows and columns to the user-item rating matrix. However, since all supplied Amazon datasets used in this analysis do not contain any item- or user-characteristic variables, we were unable to fully investigate the impact of adding those extra features to the recommender system on its prediction accuracy. Therefore, using real user and item characteristic data as extra rows and columns to the user-item rating matrix might lead to more significant improvement in recommendations.

In this paper, we combined the rating modeling technique LFM with the topic detection method LDA in order to make recommendations but we have not taken into account the fact that a word used in the textual review might have multiple senses and multiple interpretations when used in different contexts, *polysems* or that different words might actually have the same interpretations, *synonyms* [10]. In the future work we would like to redo the current experiment using *synsets* instead of *words* after applying a word sense disambiguation procedure [28]. It may be interesting to also investigate whether using sentiment analysis improves the quality of recommendations. Correspondingly, one can extend our model in such a way, that it combines the rating modeling, topic extraction, and sentiment analysis techniques for making recommendations. For instance, sentiment analysis can be used to classify whether a review is negative or positive. [42] proposed a recommender system combining rating based CF system with sentiment analysis for making recommendations. [19] introduced the joint sentiment/topic model (JST) which combines the topic modeling method LDA with sentiment analysis in order to detect a topic and a sentiment from the text simultaneously. For future work, one can try to combine the JST model with our LDA-LFM model in order to get better recommendations. In this way, one can exploit topics that carry sentiment and are possibly better proxies for our ratings.

7. REFERENCES

[1] C. Aggarwal. *Recommender Systems*. Thomas J. Watson Research Center, 2016.

[2] T. K. Aslayan and F. Flavius. Utilizing textual reviews in latent factor models for recommender systems. *36th ACM/SIGAPP Symposium on Applied Computing (SAC 2021)*, pages 1931–1940, ACM, 2021.

[3] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *28th Association for the AAAI Conference on Artificial Intelligence (AAAI 2014)*, pages 2–8. AAAI, 2014.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] P. Buono, M. F. Costabile, S. Guida, and A. Piccinno. Integrating user data and collaborative filtering in a web recommendation system. pages 315–321. Springer Berlin Heidelberg, 2002.

[6] E. Cano and M. Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis,*, 21(6):1487–1524, 2019.

[7] A. Galen and G. Jianfeng. Scalable training of l1-regularized log-linear models. In *24th International Conference on Machine Learning (ICML 2007)*, pages 33–40. ACM, 2007.

[8] T. Griffiths and M. Steyvers. Finding scientific topics. *National Academy of Sciences*, 101(1):5228–5235, 2004.

[9] J. Grivolla, D. Campo, M. Sonsona, J. Pulido, and T. Badia. A hybrid recommender combining user, item and interaction data. In *2014 International Conference on on Computational Science and Computational Intelligence (CSCI 2014)*, volume 1, pages 297–301, 2014.

[10] T. Hofmann. Probabilistic latent semantic indexing. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1999)*, pages 50–57. ACM, 1999.

[11] J. Jacoby, D. Speller, , and C. Berning. Brand choice behavior as a function of information load: Replication and extension. *Advances in Consumer Research*, 01(1):381–383, 1974.

[12] M. Kawasaki and T. Hasuike. A recommendation system by collaborative filtering including information

and characteristics on users and items. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI 2017)*, pages 1–8. IEEE, 2017.

[13] D. Kingma and L. Ba. Adam: A method for stochastic optimization. Number 1, pages 1–13. Ithaca, 2015.

[14] Y. Koren. Collaborative filtering with temporal dynamics. In *15th SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)*, pages 447–456. ACM, 2009.

[15] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[16] Y. Koren and R. Bell. *Advances in collaborative filtering. Recommender Systems Handbook*. Springer, 2011.

[17] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[18] B. Lee and W. Lee. The effect of information overload on consumer choice quality in an on-line experiment. *Psychology and Marketing*, 21(3):159–181, 2004.

[19] C. Lin and Y. He. Joint sentiment/topic model for sentiment analysis. In *18th ACM Conference on Information and Knowledge Management (CIKM 2018)*, pages 375–384. ACM, 2009.

[20] G. Ling, M. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *8th ACM Conference on Recommender Systems (RecSys 2014)*, pages 105–112. ACM, 2014.

[21] P. Lops, M. de Gemmis, and G. Semeraro. *Content-based Recommender Systems: State of the Art and Trends*. Springer, 2011.

[22] N. Lurie. Decision making in information-rich environments: The role of information structure. *Journal of Consumer Research*, 30(4):473–486, 2004.

[23] N. Malhotra. Information load and consumer decision making. *Journal of Consumer Research*, 8(4):419–430, 1982.

[24] J. McAuley and R. He. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, 1602(01585), 2016.

[25] J. McAuley and J. Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *7th ACM Conference on Recommender Systems (RecSys 2013)*, pages 165–172. ACM, 2013.

[26] J. McAuley, C. Targett, Q. Shi, and A. Van den Hengel. Image-based recommendations on styles and substitutes. *CoRR*, 1506(04757), 2015.

[27] J. Mingmin, L. Xin, Z. Huiling, and Z. Hankui. Combining deep learning and topic modeling for review understanding in context-aware recommendation. In *2018 Annual Conference of the North American Chapter of the ACL: Human Langauge Technologies (NACCL-HLT 2018)*, pages 1605–1614. ACL, 2018.

[28] R. Navigli. Word sense disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69, 2009.

[29] M. Rossetti, F. Stella, and M. Zanker. Towards explaining latent factors with topic models in collaborative recommender systems. In *24th International Workshop on Database and Expert Systems Applications (DEXA 2013)*, pages 162–167. IEEE, 2013.

[30] B. Sarwar, G. Karypis, J. Konstant, and J. Riedl. Item-based collaborative filtering recommendation algorithm. In *10th International Conference on World Wide Web (WWW 2001)*, pages 285–295. ACM, 2001.

[31] B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *1st ACM conference on Electronic commerce (EC 1999)*, pages 158–166. ACM, 1999.

[32] S. Senecal, M. Aljukhadar, and D. C.E. Information overload and usage of recommendations. In *User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI 2010)*, pages 26–33. CEUR, 2010.

[33] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009.

[34] Y. Tan, M. Zhang, Y. Liu, and S. Ma. Rating-boosted latent topics: Understanding users and items with ratings and reviews. In *25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 2640–2646. AAAI, 2016.

[35] A. Toscher, M. Jahrer, and R. Legenstein. Combining predictions for accurate recommender systems. In *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 693–702. ACM, 2010.

[36] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011)*, pages 448–456. ACM, 2011.

[37] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *6th SIAM Conference on Data Mining (SDM 2006)*, pages 549–553. SIAM, 2006.

[38] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *21st International Conference on Machine Learning (ICML 2004)*, page 116. ACM, 2004.

[39] C. Zhao, S. Suny, L. Hany, and Q. Peng. Hybrid matrix factorization for recommender systems in social networks. *Neural Network World*, 26(6):559–569, 2016.

[40] Y. Zhen, W.-J. Li, and D.-Y. Yeung. Tagicofi: Tag informed collaborative filtering. In *3rd Conference on Recommender Systems (RecSys 2009)*, pages 69–76. ACM, 2009.

[41] C. Zhu, R. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *Transactions on Mathematical Software*, 23(4):550–560, 1997.

[42] A. Ziani, N. Azizi, D. Schwab, M. Aldwairi, N. Chekkai, D. Zenakhra, and S. Cheriguene. Recommender system through sentiment analysis. In *2nd International Conference on Automatic Control, Telecommunications and Signals (ICATS 2017)*. https://hal.archives-ouvertes.fr/hal-01683511/document, 2017.

Appendix: Gradients Derivation

In this section, we present the first and second degree derivatives of the objective function with respect to the model parameters. The following equations represent the first and the second order derivatives of the proposed LDA-LFM objective function with respect to the model parameters p_u, b_u, q_i, b_i, κ and ψ, respectively.

$$\begin{aligned}
&\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial p_u} = -2q_i(r_{ui} - (\alpha + b_i + b_u + q_i^T p_u)) \\
&+ 2\lambda p_u \\
&= -2q_i(r_{ui} - \hat{r}_{ui}) + 2\lambda p_u \\
&= -2q_i e_{ui} + 2\lambda p_u \\
&= -2(q_i e_{ui} - \lambda p_u)
\end{aligned}$$

$$\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial p_u \partial p_u} = 2\lambda \tag{18}$$

$$\begin{aligned}
&\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial b_u} = -2(r_{ui} - (\alpha + b_i + b_u + q_i^T p_u)) + 2\lambda b_u \\
&= -2(r_{ui} - \hat{r}_{ui}) + 2\lambda b_u \\
&= -2e_{ui} + 2\lambda b_u \\
&= -2(e_{ui} - \lambda b_u)
\end{aligned}$$

$$\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial b_u \partial b_u} = 2\lambda \tag{19}$$

$$\begin{aligned}
&\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial b_i} = -2(r_{ui} - (\alpha + b_i + b_u + q_i^T p_u)) + 2\lambda b_i \\
&= -2(r_{ui} - \hat{r}_{ui}) + 2\lambda b_i \\
&= -2e_{ui} + \lambda b_i \\
&= -2(e_{ui} - \lambda b_i)
\end{aligned}$$

$$\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial b_i \partial b_i} = 2\lambda \tag{20}$$

$$\begin{aligned}
&\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial \kappa} = -\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} q_{d,z_{d,j}} \\
&- \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})} q_{d,z_{d,j}} \Big) \\
&= -\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} q_{d,z_{d,j}} \big(1 - \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})}\big) \Big)
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial \kappa \partial \kappa} = \mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} q_{d,z_{d,j}} \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}}))} \\
&\big(1 - \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}}))}\big) \Big)
\end{aligned} \tag{21}$$

$$\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial \psi} = -\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} 1 - \frac{exp(\psi_{z_{d,j}, w_{d,j}})}{\sum_{w'} exp(\psi_{z_{d,j}, w'})} \Big)$$

$$\begin{aligned}
&\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial \psi \partial \psi} = \mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} \frac{exp(\psi_{z_{d,j}, w_{d,j}})}{\sum_{w'} exp(\psi_{z_{d,j}, w'})} \\
&\big(1 - \frac{exp(\psi_{z_{d,j}, w_{d,j}})}{\sum_{w'} exp(\psi_{z_{d,j}, w'})}\big) \Big)
\end{aligned} \tag{22}$$

$$\begin{aligned}
&\frac{\partial f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial q_i} = \frac{-2}{\mid \mathcal{T} \mid} \sum_{u,i \in \mathcal{T}} \big((r_{ui} - \hat{r}_{u,i}) p_u + 2\lambda_i q_i\big) - \\
&\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} \kappa - \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})} \kappa \Big) \\
&= \frac{-2}{\mid \mathcal{T} \mid} \sum_{u,i \in \mathcal{T}} \big((r_{u,i} - \hat{r}_{u,i}) p_u + 2\lambda q_i\big) - \\
&\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} \kappa \big(1 - \frac{exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})}\big) \Big)
\end{aligned}$$

$$\begin{aligned}
&\frac{\partial^2 f(\mathcal{T} \mid \Theta, \Phi, \kappa, z)}{\partial q_i \partial q_i} = \frac{-2}{\mid \mathcal{T} \mid} \sum_{u,i \in \mathcal{T}} 2\lambda - \\
&\mu\Big(\sum_{d \in \mathcal{T}} \sum_{j=1}^{N_d} \frac{\kappa^2 exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})} \big(1 - \frac{\kappa^2 exp(\kappa q_{d,z_{d,j}})}{\sum_{k=1}^{K} exp(\kappa q_{d,z_{d,k}})}\big) \Big)
\end{aligned} \tag{23}$$