



Recommendation systems for online advertising

Sumit Sidana

► To cite this version:

Sumit Sidana. Recommendation systems for online advertising. Computers and Society [cs.CY]. Université Grenoble Alpes, 2018. English. NNT : 2018GREAM061 . tel-02060436

HAL Id: tel-02060436

<https://theses.hal.science/tel-02060436>

Submitted on 7 Mar 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Sumit SIDANA

Thèse dirigée par **Massih-Reza AMINI**, Professeur, Université Grenoble Alpes

préparée au sein du **Laboratoire Laboratoire d'Informatique de Grenoble**
dans l'**École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Filtrage collaboratif en ligne : application à la publicité programmatique

Dynamic collaborative filtering for on-line advertising

Thèse soutenue publiquement le **8 novembre 2018**,
devant le jury composé de :

Monsieur MASSIH-REZA AMINI

PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Directeur de thèse

Madame CHARLOTTE LACLAU

MAITRE DE CONFERENCES, UNIVERSITE JEAN MONNET - SAINT-ETIENNE, Co-directeur de thèse

Monsieur PATRICK GALLINARI

PROFESSEUR, SORBONNE UNIVERSITES - PARIS, Rapporteur

Madame JOSIANE MOTHE

PROFESSEUR, UNIVERSITE TOULOUSE-JEAN JAURES, Rapporteur

Madame SIHEM AMER-YAHIA

DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES, Président

Monsieur ROMARIC GAUDEL

PROFESSEUR ASSISTANT, ENSAI - RENNES, Examineur

Monsieur GILLES VANDELLE

RESPONSABLE SCIENTIFIQUE, SOCIETE KELKOO - ECHIROLLES, Examineur



Acknowledgments

First of all, I would like to express my gratitude to my supervisor Massih-Reza Amini and co-supervisor Charlotte Laclau. I am grateful for your guidance, patience, advice, time-involved, and for the research ideas shared with me during these three years. The encouragement, guidance and resources you provided helped me to continue my research and finish this thesis.

Then, I would like to thank Patrick Gallinari and Josiane Mothe, who kindly agreed to review my Ph.D. thesis. I would also like to thank Gilles Vandelle and Romaric Gaudel, who agreed to be part of my thesis committee.

Thanks to Sihem Amer-Yahia of CNRS, who first gave me an opportunity of internship in LIG, where I got my first research experience, and which encouraged me to start a PhD. I am also grateful to you for agreeing to be part of my thesis committee.

Thank you to my parents for their emotional support and encouragement during this PhD.

Thanks to all the AMA members (past and present) for their help and support: Bikash, Sami, Parantapa, Hamid, Yagmur, Vera, Karim, Thibaut, Anil, Julien, Myriam, Georgios, Adrien, Maziar, Hesam, Lauren, Saeed, Saeid and Vasilii.

Thanks to LIG and Université Grenoble Alpes for their financial and logistical support. Thanks to FUI for their financial participation in this thesis.

Thanks to Kelkoo and Purch engineers for providing data and advice in this PhD.

Abstract

This thesis is dedicated to the study of Recommendation Systems for implicit feedback (clicks) mostly using Learning-to-rank and neural network based approaches. In this line, we derive a novel Neural-Network model that jointly learns a new representation of users and items in an embedded space as well as the preference relation of users over the pairs of items and give theoretical analysis. In addition we contribute to the creation of two novel, publicly available, collections for recommendations that record the behavior of customers of European Leaders in eCommerce advertising, Kelkoo¹ and Purch². Both datasets gather implicit feedback, in form of clicks, of users, along with a rich set of contextual features regarding both customers and offers. Purch's dataset, is affected by popularity bias. Therefore, we propose a simple yet effective strategy on how to overcome the popularity bias introduced while designing an efficient and scalable recommendation algorithm by introducing diversity based on an appropriate representation of items. Further, this collection contains contextual information about offers in form of text. We make use of this textual information in novel time-aware topic models and show the use of topics as contextual information in Factorization Machines that improves performance. In this vein and in conjunction with a detailed description of the datasets, we show the performance of six state-of-the-art recommender models.

Keywords. Recommendation Systems, Data Sets, Learning-to-Rank, Neural Network, Popularity Bias, Diverse Recommendations, Contextual information, Topic Model.

¹<https://www.kelkoo.com/>

²<http://www.purch.com/>

Contents

Contents	v
1 Introduction	3
1.1 Challenges in Online Adverstising	4
1.2 Contributions	6
1.3 Thesis structure	7
2 Recommender Systems: state-of-the-art and evaluation	9
2.1 Definition of Personalized Recommendation	10
2.2 Content Based Recommender Systems	11
2.3 Collaborative Filtering	11
2.3.1 Memory-based CF	12
2.3.2 Matrix Factorization and Low-Rank Approximation	13
2.3.3 Factorization Machines	18
2.4 Collaborative Ranking	20
2.4.1 Learning-to-Rank	20
2.4.2 Pairwise-Ranking for Recommendation Systems	22
2.5 Deep Learning for Recommender Systems	24
2.5.1 Representation Learning (RL) with Embeddings	25
2.5.2 Users and Items Representation Learning (RL) with Embeddings	27
2.6 Diversity in Recommender Systems	28
2.7 Evaluation of Recommender Systems	30
2.7.1 Prediction Accuracy	30
2.7.2 Ranking Measures	32
2.7.3 Diversity Measures	33

2.7.4	Online-Testing	34
2.8	Conclusion	35
3	Data-collections	37
3.1	Introduction	38
3.2	Collection of the data	38
3.3	KASANDR Dataset	41
3.3.1	Structure of the data	41
3.3.2	Basic statistics	42
3.4	PANDOR Dataset	45
3.4.1	Structure of PANDOR	45
3.4.2	Features of PANDOR	46
3.4.3	Summary	48
4	Extracting latent topics over timely related articles	49
4.1	Introduction	50
4.2	General-purpose topic modelling	51
4.2.1	Latent Dirichlet Allocation (LDA)	51
4.2.2	Topic-Aspect Model (TAM)	53
4.3	Temporal Latent Topic Models	54
4.3.1	Temporal-LDA (TM-LDA)	54
4.3.2	Time-Aware Topic-Aspect Model	55
4.4	Application to health monitoring on social media over time	58
4.4.1	TM-LDA applied to health documents	59
4.4.2	T-ATAM	59
4.5	Results	60
4.5.1	Data	60
4.5.2	Comparison between models	61
4.6	Conclusion	61
5	Jointly Learning embeddings and user preference through implicit feed-back	63
5.1	Introduction	64
5.2	Theoretical Study	64

5.3	A Neural Network model to learn user preference	71
5.4	Diversity	76
5.4.1	Incorporating diversity to handle popularity bias in recommender systems	76
5.5	Conclusion	77
6	Experimental Results	79
6.1	Introduction	80
6.2	Baselines and Evaluation Protocol	80
6.3	NERvE Results	83
6.4	Results on KASANDR and PANDOR	90
6.5	Conclusion	99
7	Conclusions and future perspectives	101
	List of publications	107
	References	109

Notations

$\mathcal{U} \subseteq \mathbb{N}$; $\mathcal{U} = \{u_1, \dots, u_n\}$ is the set of n users or the set of indexes over users

$\mathcal{I} \subseteq \mathbb{N}$; $\mathcal{I} = \{i_1, \dots, i_m\}$ is the set of m items or the set of indexes over items

R = a sparse preference matrix of size $n \times m$

r_{ui} = Rating given by user u for item i

\hat{r}_{ui} = Predicted rating of target user u of item i

p_u = Latent feature vector of user u

q_i = Latent feature vector of item i

(i, u, i') = A triplet composed by the indexes of an item i , a user u and a second item i'

\succ_u = Preference relation

\mathbf{U}_u = The transformed embedding vector of user u

\mathbf{V}_i = The transformed embedding vector of item i

$\tilde{\mathcal{S}}_n$ = A random set of size n of interactions by building triplets (i, u, i')

$\mathfrak{N}_{u,k}$ = A ranked list of the $k \ll M$ preferred items for each user in the test set

\mathcal{S}_u^k = the list of items and k its size

$\mathbf{V}_i^{\ell_1}$ (resp. $\mathbf{V}_{i'}^{\ell_1}$) = the ℓ_1 -normalized embedding associated with item i (resp. i')

β is the diversity inducing regularization parameter whose role is to induce more or less diversity in the final list of recommended items

\mathcal{P} = Posts

\mathcal{G} = Regions

\mathcal{T} = Time periods

\mathcal{P}_g^t = Posts from region g during time t

D_g^t = Document-set built by mapping the content of each post $p \in \mathcal{P}_g^t$ to a document

Chapter 1

Introduction

In the recent years, recommender systems (RS) have attracted a lot of interest in both industry and academic research communities, mainly due to new challenges that the design of a decisive and efficient RS presents. Given a set of customers (or users), the goal of RS is to provide a personalized recommendation of products to users which would likely to be of their interest. This process is described in Figure 1.1. Common examples of applications include the recommendation of movies (Netflix, Amazon Prime Video), music (Pandora), videos (YouTube), news content (Outbrain) or advertisements (Google). The development of an efficient RS is critical from both the company and the consumer perspective. On one hand, users usually face a very large number of options: for instance, Amazon proposes over 20,000 movies in its selection, and it is therefore important to help them to take the best possible decision by narrowing down the choices they have to make. On the other hand, major companies report significant increase of their traffic and sales coming from personalized recommendations: Amazon declares that 35% of its sales is generated by recommendations, two-thirds of the movies watched on Netflix are recommended and 28% of ChoiceStream users said that they would buy more music, provided the fact that they meet their tastes and interests.¹

This thesis is part of the FUI project Calypso, which is designed with the main goal of improving the performance of the e-commerce advertisements, that will be generating a large part of the income of the partner companies, namely Kelkoo and

¹Talk of Xavier Amatriain - Recommender Systems - Machine Learning Summer School 2014 @ CMU.

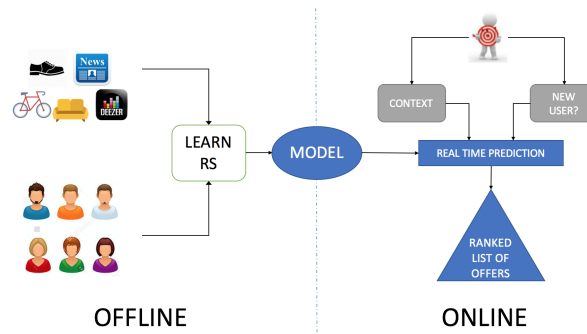


Figure 1.1 – A model is learned based on the set of available user preferences over items. It is then used to recommend new items to the same set of users (or new users).

Purch. With the help of predictive machine learning approaches, the project proposes to increase the probability of clicking on the products presented to Internet users in advertising inserts on the sites of both partners and purchased on advertising market places. Internet advertising has become a major economic challenge for online selling companies that have to optimize their catalogs in real time, in order to propose the products to users that fit the best to their interests and preferences. The companies are required to display advertisements and offers which user may be interested in engaging and buying. The overall model needs to address several challenges including but not limited to, being scalable in order to deal with a large amount of data, work with highly sparse and implicit feedback, and being able to handle heterogeneous contextual information regarding both the users and the products.

1.1 Challenges in Online Adverstising

While building efficient and scalable models for online advertising, there are various challenges and issues which need to be addressed. These are as follows.

Types of feedback

The majority of approaches for RS are based on the previous feedback given by the users of the system. These feedback can be of different nature when items are presented to her/him, and we broadly classify them as being either implicit or explicit.

Explicit feedback is probably the most common one in the literature, and can be in

1. INTRODUCTION

the form of ratings¹, up-votes or likes, for instance. However, while explicit feedback generally provides a relevant signal for making recommendation, it is also much more difficult to gather than implicit one, as one needs to convince the user to rate or explicitly tell their preference after having consumed the item. For this reason, feedback used for building efficient RS has evolved over time, from explicit feedback to mostly implicit one.

Implicit feedback is usually inferred from user's behavior while interacting with the system, and include clicking on items, bookmarking a page or listening to a song. Implicit feedback, as contrary to explicit feedback, is in abundance and does not require an extra effort from user's side. However, it also presents several challenging characteristics. Firstly, implicit feedback does not clearly depict the preference of a user for an item; for instance a user listening to a song or clicking on a product does not necessarily mean that he or she likes the corresponding item. For this reason, one cannot measure the degree of preference from such type of interactions. Secondly, these data present a scarcity of negative feedback, i.e., only positive observations. This is because a user not clicking on a product may be because of various reasons, such as, lack of time, or the offer not being in display region of the banner. For this reason, considering the lack of any positive signal as a negative signal introduces bias in building models. There are no true negatives in implicit feedback.

However, because of its presence in abundance and availability, research on implicit feedback has gained increasing attention in very recent years [He et al., 2016] and have been greatly encouraged by competitions organized by some of the major industrial actors, like Criteo², Outbrain³, or Spotify⁴, for instance.

Sparsity

Given large sets of users and items, sparsity arises from the fact that users in general rate or click only a very limited number of items, compared to the number of items available in the catalogue that are shown to them. This problem is extremely common in recommender systems, see for example the degree of sparsity present in RS bench-

¹<https://www.netflixprize.com/>

²<https://www.kaggle.com/c/criteo-display-ad-challenge>

³<https://www.kaggle.com/c/outbrain-click-prediction>

⁴<http://www.recsyschallenge.com/2018/>

marks shown in Table 1.1, and needs to be taken into account while designing a RS. In addition, for data extracted from online advertising, which interests us in this thesis, this phenomenon is even more pronounced.

Table 1.1 – Statistics of various collections used in our experiments after preprocessing.

	# of users	# of items	# of interactions	Sparsity
ML-100K	943	1,682	100,000	93.685%
ML-1M	6,040	3,706	1,000,209	95.530%
NETFLIX	90,137	3,560	4,188,098	98.700%
KASANDR	25,848	1,513,038	9,489,273	99.976%
PANDOR	1,918,968	3,755	225,579	99.997%

Popularity-Bias

Finally, popularity-bias is another prevalent issue in RS and raises the question of *how* we recommend instead of just *what* we recommend to the users.

Indeed, the goal of a RS is to have fewer flops on the top [Paudel et al., 2017] of the recommended list, and inducing more diversity in this recommended list ensures that user may prefer to interact with at least some items in contrast to the situation where we introduce just monotonous relevant items. In addition, the recent work of [Abdollahpouri et al., 2017] shows that diversity can be used in order to control the popularity bias in such type of data, also known as the problem of long tail i.e. a situation where a large majority of items have only very few ratings or clicks, either because they are new or just unpopular. In recent works [Herlocker et al., 2004; McNee et al., 2006], it has been shown that only recommending relevant items (within the semantics of items which are the more likely to be of interest for users) has its limits and that adding notions of diversity and discovery in the process can highly increase the performance of a recommendation system [Bradley and Smyth, 2001; McSherry, 2002; Smyth and McClave, 2001; Zhang and Hurley, 2008].

1.2 Contributions

In this thesis, we tried to address these challenges by exploring different type of RS models.

These models are mainly evaluated on the basis of two newly created datasets, KASANDR and PANDOR, arising from the real online traffic recorded by the two partners of the project, and that we made publicly available to the research community, along with a detailed description. We show that PANDOR, in particular, suffers from popularity bias and propose diversity based approach, which are based on regularization, in order to handle diversity.

Then, in order to deal with the textual content in that we had in hand, we develop two novel time-aware topic modelling approaches, which are able to extract latent topics in temporally sequenced textual data. We show that topics derived from this time-aware topic model can be used to improve the performance of RS models.

Furthermore, for the personalized recommendation part, we propose a novel neural-network based architecture which can handle data sparsity by learning both a good dense representation of users and items, as well as the ranked list of the preferences for all users. This model, in addition to be efficient with implicit feedback, also allows to deal with large datasets and can easily integrate contextual information, of diverse nature.

1.3 Thesis structure

The rest of this thesis is organized into 5 chapters. The main contents of each chapter are summarized below:

Chapter 2 : Describes some common state-of-the-art approaches developed in RS. We mainly focused on collaborative filtering, learning to rank, item-embedding and diversity based approaches for recommender systems as our contributions also build upon these ideas.

Chapter 3 : Presents in detail large scale data sets we have contributed to RS community during the discourse of this thesis. In particular, we describe basic characteristics of KASANDR and PANDOR and how making them public can help RS research community to benchmark their approaches and models on these data sets.

Chapter 4 : Details the topic-modelling techniques to extract evolution of latent concepts with time from textual data and how they can be applied in context of RS. We first study general purpose topic modeling techniques. Then, we present two novel time-aware topic models. We study these approaches as an application to health monitoring and run experiments to show how our models outperform the existing techniques to predict the evolution of topics with time.

Chapter 5 : Presents a neural network to optimize two loss functions simultaneously in order to come up with better representation and ranking functions. In particular, we learn embedding based representations and pairwise ranking function by optimizing two losses simultaneously. We then extend this neural network to recommend, not only relevant offers, but diverse offers as well.

Chapter 6 : Sets out the experiments we conducted in order to show the efficiency of our models. We first study various settings of the parameters and their effects on performance of the neural network we developed. Then, we go on to show that it outperforms the existing techniques in recommending offers when the prediction function is learned using implicit feedback. Then, we give benchmark results on popular RS algorithms which have been known to perform well on implicit feedback on KASANDR. Then, we run experiments on PANDOR and present results of various baselines. We show that baselines run on PANDOR suffer from popularity bias and performance of models can be improved by using diversity based regularizers. We also show the results of the models can be improved by using topics from time-aware topic models.

Chapter 2

Recommender Systems: state-of-the-art and evaluation

2.1 Definition of Personalized Recommendation

Personalized recommendations consist of selecting products or offers from the catalog that create a relevant, individualized interaction environment designed to enhance the experience of the user. It uses insight based on the user’s personal data, as well as behavioral data about the actions of similar individuals, to deliver an experience that meets specific needs and preferences. Advantage of using personalized approaches to recommendation is that these approaches generally outperform non-personalized counterparts in performance and are interesting both from academic and industry point of view.

Before the breakthrough of ML in RS, recommendations were made based on non-ML approaches, such as for instance, random-based approach, consisting of recommending random items to a given user, or popularity-based approaches consisting of recommending the most popular items to all users. While these type of approaches are usually outperformed by personalized recommendation methods, they can still be used to deal with specific challenges such as user and item cold-start.

In this chapter, we give a brief overview of three families of RS models that are arguably the most widely used for our task, notably: collaborative filtering, learning-to-rank and deep learning. The remainder of this chapter is organized as follows. Section 2.2 describes the content filtering approaches principle, their advantages and problems. Section 2.3 presents the general idea behind collaborative filtering (CF). We first define CF and then categorize them into memory-based CF and latent factor models. Specifically, we describe Matrix Factorization, Factorization Machines. Then, in section 2.4, we discuss ranking-based CF approaches in detail. We discuss how pairwise learning-to-rank has been applied to recommender systems. In Section 2.5, we give details of how deep learning approaches have been successfully applied to RS. We discuss the concept of representation learning in detail. Finally, in section 2.7, we discuss various evaluation approaches widely used in RS and specifically the ones which are relevant to our contributions.

2.2 Content Based Recommender Systems

Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties [Mooney and Roy, 1999]. These approaches present numerous advantages as we are assisted by an increased availability of content information and semantic relationship data, through social tagging, reviews, platform like BabelNet or Wikipedia. Perhaps, the first popular content based recommender system was built by [Kamba et al., 1996], where the system architects built a personalized news recommender system.

Content-based recommender systems can be broadly classified in two ways in order to recommend on the basis of content (product attributes). Firstly, long term techniques consist of building profile of content preferences. Secondly, content based techniques are also good at helping users browse through catalogs/baskets. For example, while purchasing things at Amazon, usually we are shown the items similar to the items we already have in our basket. Content filtering works by first building profile of each item by using TF-IDF (documents), meta-data (movies) or tags (images). Then, user profiles are built by aggregating profiles of items rated or consumed by them. Unrated items are then evaluated by taking the dot-product between item-profile and user-profile.

The biggest issues with content-based recommender systems is that item-profiles have to be built and good domain knowledge of items is required, which is not always feasible. Additionally, user cold-start problem cannot be solved by content-based recommender systems. For more details on content-based recommender systems, we refer the reader to the surveys of [Lops et al., 2011; Pazzani and Billsus, 2007].

2.3 Collaborative Filtering

Collaborative Filtering are the set of techniques which ignore user and item attributes but focus on user-item interactions. They are the pure behavior based recommendation techniques. Traditionally, we distinguish between memory-based and model-based approaches. The latter approach is arguably the most popular one nowadays, and in this section we focus on three of them, including Matrix Factorization, Ranking CF

and Factorization Machines, which are of particular interest to us in this thesis.

2.3.1 Memory-based CF

Memory-based techniques use the data (likes, votes, clicks, etc) that you have to establish correlations between either users or items to recommend an item i to a user u who has never seen it before. In the case of user-based approach, we get the recommendations from items seen by the users who are closest to u . In contrast, item based approach tries to compare items using their characteristics (movie genre, actors, books publisher or author etc) to recommend similar new items ¹.

User-user CF (UUCF) is the most commonly used form of personalized memory-based CF [Herlocker et al., 1999]. In order to predict which items should be displayed or recommended to a given user, the system relies on the analysis of the neighborhood of this particular user. This neighborhood is composed based on past interactions, and include other users who have presented similar taste for other items.

More formally, given a set of items $\mathcal{I} \subseteq \mathbb{N}$, and a set of users $\mathcal{U} \subseteq \mathbb{N}$, and a sparse matrix of ratings R , we compute the prediction \hat{r}_{ui} as follows:

- For all users $v, u \in \mathcal{U}$, such that $u \neq v$, compute w_{uv} (a similarity metric - eg. Pearson correlation coefficient)
- Select a neighborhood of users $N_k \in \mathcal{U}$ with highest w_{uv}
 - May limit the neighborhood to top-k neighbours
 - May limit neighborhood to $w_{uv} > \epsilon$, where ϵ is a similarity threshold.
- Compute prediction

User-user CF suffers from sparsity issues. With large item-sets and small number of ratings or clicks, too often, there are points where no recommendations for a user, who doesn't have common ratings with other users, can be made. Many solutions have been proposed to address this problem, with item-item Collaborative filtering being the most common one.

¹<https://yasserebrahim.wordpress.com/2012/10/13/memory-based-vs-model-based-recommendation-systems/>

Item-item collaborative filtering (IICF) was first introduced by [Sarwar et al., 2001] and overcomes sparsity and computational issues of UUCF in areas where $m \gg n$ (m : number of users, n : number of items). Item-item similarity is on the items which are co-rated and can also be used to directly recommend top-k items [Deshpande and Karypis, 2004] in the case of implicit feedback.

While memory-based approaches were the the first ones present in commercial RS, they present numerous drawbacks, which are as follows. Firstly, the RS datasets suffer from sparsity. Evaluation of RS systems goes through large item sets and users' interactions on these item sets are under 1%. There is a poor relationship among like minded but sparse-rating users and memory-based CF fail to capture similarity between such users. Secondly, it is difficult to make predictions based on nearest neighbor algorithms and accuracy of recommendation may be poor. Thirdly, scalability is an issue with memory-based CF. Computation of nearest neighbor requires computation that grows with both the number of users and the number of items. Instead of using all previous ratings to make a prediction, model-based approaches first build a model from theses ratings, and use this model to make further recommendations. In what follows, we describe some popular model-based methods which have established themselves as main baselines over the past years.

2.3.2 Matrix Factorization and Low-Rank Approximation

Principle of Matrix Factorization Ratings can be seen as matrix or dyadic representation as shown in Figure 2.1. But, ratings matrix is an over fit representation of user tastes and item attributes. This rating matrix can actually be seen as the product of lower dimensional matrices representing user tastes and item attributes. This idea gives rise to notion of factorizing matrix into lower dimensional matrices.

Formally, we aim to find the users and items feature matrices, denoted by \mathbf{U} and \mathbf{V} , respectively, minimizing the squared error over the known ratings,

$$(\mathbf{U}, \mathbf{V}) = \arg \min_{\mathbf{U}, \mathbf{V}} \sum_{\forall (u,i) \in \mathcal{S}} (r_{ui} - \mathbf{U}_u \mathbf{V}_i^T)^2,$$

where $\mathbf{U} \in \mathbb{R}^{n \times k}$, $\mathbf{V} \in \mathbb{R}^{m \times k}$, are latent representations of users and items, defined in a lower-dimensional space, i.e., $k \ll \min(n, m)$. However, since a large part of

the ratings in the matrix are unknown, one usually considers the following regularized optimization problem (\mathcal{S} is the set of observed ratings)

$$(\mathbf{U}, \mathbf{V}) = \arg \min_{\mathbf{U}, \mathbf{V}} \sum_{(u,i) \in \mathcal{S}} (r_{ui} - \mathbf{U}_u \mathbf{V}_i^T)^2 + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2), \quad (2.3.1)$$

where λ controls the extent of regularization to avoid overfitting, and can be determined by cross-validation.

To proceed, we introduce the main optimization approaches which have been proposed in order to solve this minimization problem, over the past years.

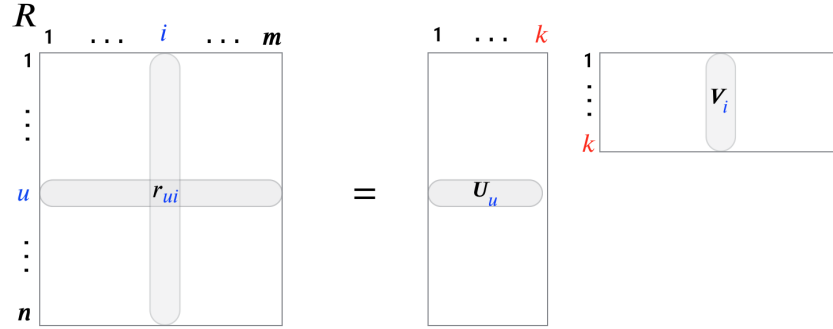


Figure 2.1 – Principle of Matrix Factorization.

Optimization approaches

We can broadly distinguish three main approaches, which propose to optimize the problem defined above: the Singular Value Decomposition (SVD), Stochastic Gradient Descent (SGD) and Alternating Least Squares (ALS).

Singular Value Decomposition (SVD) proposes to decompose a given matrix $R \in \mathbb{R}^{n \times m}$ as follow

$$R = P \Sigma Q^T,$$

where P and Q are two squared orthogonal matrices of size n and m , respectively, and Σ is a diagonal matrix of size $n \times m$ that contains the non-increasing and non-negative singular values of R . SVD presents several interesting properties, including the fact

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

that it can be used on any matrix which contains real entries, and one can show that it gives the best rank-k approximation of the original ratings matrix under the global root mean squared error (RMSE), meaning that this approximation is optimal in the Frobenius norm.

However, SVD also admits import downsides, especially in the context of Recommender Systems. Firstly, decomposing the ratings matrix is slow, which can be an important issue as in RS, one need to handle an extremely large volume of data in a very limited amount of time. Secondly, SVD can only be applied on complete matrix, meaning that one need to know the preferences of all users for all the movies in advance, a situation that will of course make RS unnecessary. To overcome the latter, some imputation strategies have been proposed, such as imputing the mean rating of items or even null values [Sarwar et al., 2000]. While this strategy allows to obtain a complete matrix, it also introduces an important bias in the data and requires to deal with a larger amount of (superficial) data.

Stochastic gradient descent(SGD) The optimization of Equation 2.3.1 using Stochastic Gradient Descent (SGD) was first popularized by [Funk, 2006]. The algorithm first initializes the user and item latent matrices, \mathbf{U} and \mathbf{V} , then loops through all ratings in the training set. For each training instance, the following prection error is computed:

$$e_{ui} = r_{ui} - \mathbf{V}_i^\top \mathbf{U}_u.$$

Then, based on this prediction error, the latent features \mathbf{V}_i and \mathbf{U}_u are modified in the opposite direction of the gradient in the following manner:

$$\mathbf{V}_i \leftarrow \mathbf{V}_i + \gamma \cdot (e_{ui} \cdot \mathbf{U}_u - \lambda \cdot \mathbf{V}_i),$$

$$\mathbf{U}_u \leftarrow \mathbf{U}_u + \gamma \cdot (e_{ui} \cdot \mathbf{V}_i - \lambda \cdot \mathbf{U}_u),$$

where γ is the learning rate of the gradient descent, and λ is the regularization parameter defined above. Then, the prediction is made using the updated \mathbf{U}_u and \mathbf{V}_i . This process of updating the parameters and predicting the rating using updated parameters keeps going on until a *fixed number of iterations* or if the error e_{ui} is below a specific *threshold*. The number of iterations or the threshold is usually fixed by cross-

validation. SGD based approach is both easy in implementing and has a fast running time [Koren, 2008; Paterek, 2007; Takács et al., 2007]. Indeed, if we set the number of epochs to T , and k is the dimension size of \mathbf{V}_i and \mathbf{U}_u , then the time complexity of the SGD procedure is $O(Nk)$.

Alternating least squares (ALS) Alternating Least Squares (ALS), initially proposed by [Jain et al., 2013], relies on an iterative optimization procedure that consists of the two following steps

1. Fix the item latent matrix \mathbf{V} and solve the quadratic equation (see Eq. 2.3.1) for the user latent matrix \mathbf{U} , i.e.

$$\mathbf{U}_u = (\sum_{(u,i) \in \mathcal{S}} \mathbf{V}_i \mathbf{V}_i^\top + \lambda \mathbf{Id})^{-1} \sum_{(u,i) \in \mathcal{S}} r_{ui} \mathbf{V}_i. \quad (2.3.2)$$

2. Fix the user latent matrix \mathbf{U} and solve the same quadratic equation, this time for the item latent matrix \mathbf{V} , i.e.

$$\mathbf{V}_i = (\sum_{(u,i) \in \mathcal{S}} \mathbf{U}_u \mathbf{U}_u^\top + \lambda \mathbf{Id})^{-1} \sum_{(u,i) \in \mathcal{S}} r_{ui} \mathbf{U}_u. \quad (2.3.3)$$

where \mathbf{Id} is the identity matrix. As for SGD, the algorithm alternates between these two steps until convergence or for a number of iterations given in advance. While SGD is faster than ALS, still ALS is desirable in couple of cases. On the one hand, ALS computes each \mathbf{V}_i independently of the other item factors, and each \mathbf{U}_u independently of the other user factors, giving rise to potentially massive parallelization of the algorithm [Zhou et al., 2008]. On the other hand, ALS is also more preferable in the case of implicit datasets; because the training set cannot be considered sparse, looping over each single training case as gradient descent does, is not practical [Hu et al., 2008a].

Other formulations of Matrix Factorization

Adding bias : There are systematic biases present in ratings. For example, some users are generous and tend to give higher ratings than others. Likewise, some items tend to get higher ratings than others as they are more popular and perceived in a better

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

way. To address these issues the Equation 2.3.1 provides fairly simple way of incorporating such biases. The system then minimizes the following objective function:

$$\arg \min_{\mathbf{U}, \mathbf{V}, \mathbf{b}} \sum_{u,i} (r_{ui} - \mu - b_u - b_i - \mathbf{V}_i^\top (\mathbf{U}_u))^2 + \lambda (\|\mathbf{U}_u\|^2 + \|\mathbf{V}_i\|^2 + b_u^2 + b_i^2),$$

where μ is the overall average rating; $\mathbf{b} = (b_u, b_i)$ contains the deviations of user u and item i from μ , respectively.

Next, we present another version of this model, which, by adding variables, can be optimized for implicit preferences.

Matrix Factorization for implicit feedback All the above matrix factorization methods have been used for learning latent user and item factors from explicit feedback. The traditional model with bias was then extended by [Koren, 2008], where an extra term was added in for incorporating implicit feedback as follows:

$$\arg \min_{\mathbf{U}, \mathbf{V}, \mathbf{b}} \left(\sum_{u,i} r_{ui} - \mu - b_u - b_i - \mathbf{V}_i^\top (\mathbf{U}_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j) \right)^2 + \lambda (\|\mathbf{U}_u\|^2 + \|\mathbf{V}_i\|^2 + b_u^2 + b_i^2),$$

where $N(u)$ is the set of items for which user u expressed implicit preference (e.g. click, like); $\sum_{j \in N(u)} y_j$ is the vector for a user u who showed a preference for items in $N(u)$; finally, b_u, b_i are the biases introduced in Equation 2.3.2.

[Hu et al., 2008c] also came up with a novel version of MF able to handle for implicit feedback. To proceed, they propose the following objective function in their formulation:

$$\arg \min_{\mathbf{U}, \mathbf{V}, \mathbf{b}} \sum_{u,i} c_{ui} (r_{ui} - \mathbf{V}_i^\top \mathbf{U}_u)^2 + \lambda \left(\sum_u \|\mathbf{U}_u\|^2 + \sum_i \|\mathbf{V}_i\|^2 \right),$$

where c_{ui} means the extent to which we penalize the error on user u on item i . The standard choice for c_{ui} in the explicit feedback case is $c_{ui} = 1$, if $(u, i) \in \mathcal{S}$ and 0 otherwise, where \mathcal{S} are the set of observed ratings. While Matrix Factorization approaches are the most commonly used approach for RS, it is difficult to use contextual

information along with such approaches. Factorization Machines, which we discuss next, overcome this drawback.

2.3.3 Factorization Machines

Factorization machines (FM) are a generic approach that allows to mimic most factorization models by feature engineering. This way, FM combine the generality of feature engineering with the superiority of factorization models in estimating interactions between categorical variables of large domain¹. FM [Rendle, 2010] can be seen as a hybrid solution between classification approaches (such as SVM) and factorization approaches (such as matrix factorization). FM is known to handle very high sparsity, runs in linear time and can handle contextual information. Matrix Factorization can be shown to be just the special case of FM.

Let us consider the simple example presented in Figure 2.2, where we have

$$\mathcal{S} = (Tom, Book, 1), (Tom, Movie, 3), (Jack, Music, 3), \dots, (Alice, Music, 2)$$

Factorization machines are the general predictors like SVM working with real valued feature vector. FM relies on a specific feature representation, which differs from the classic dyadic representation User-Items: each user, item and contextual information is transformed into a real valued feature vector with the corresponding target. In Figure 2.2, there are first $|U|$ binary indicator variables (blue), that represent active user of transaction. The next $|I|$ binary indicator variables hold the active item. Then, there are other user and item features which may be real valued.

FM break the independence of interaction parameters by factorizing them. The prediction function of FM is given by:

$$f(x) := \underbrace{w_0}_{\text{bias}} + \underbrace{\sum_{i=1}^n w_i x_i}_{\text{Linear Regression}} + \underbrace{\sum_{i=1}^n \sum_{j>i}^n \sum_k v_{ik} v_{jk}}_{\text{factorization}} \underbrace{\langle x_i, x_j \rangle}_{\text{interaction}} \quad (2.3.4)$$

¹<http://www.libfm.org/>

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

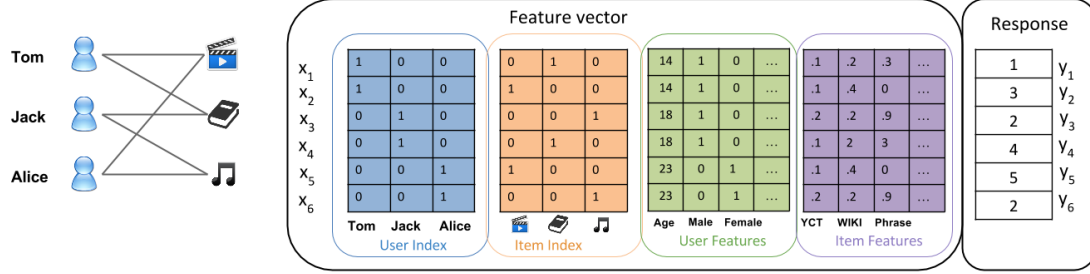


Figure 2.2 – Factorization Machines : from a dyadic representation to a new feature space (taken from Minchul Kim, 2017).

The model parameters that have to be estimated in equation 2.3.4 are:

$$\Theta = \{w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}, \mathbf{V} \in \mathbb{R}^{n \times k}\}$$

A row \mathbf{v}_i within \mathbf{V} describes the i -th variable with k factors. Then, [Rendle et al., 2011] extended their model to handle context as well. Another significant extension of FM came in the form of Field-aware factorization machines [Juan et al., 2016b] where authors used different latent factors for different feature parameter pairs.

$$f(x) := \sum_{j_1=1}^n \sum_{j_2=j_1+1}^n (\mathbf{w}_{j_1, f_2} \cdot \mathbf{w}_{j_2, f_1}) x_{j_1} x_{j_2}.$$

There are various disadvantages of using FM/FFM. Firstly, performance of FM/FFM depends on features in data, and as we keep adding features to FM/FFM to increase performance, the running time keeps getting slow. To overcome this drawback a lighter version of FFM was developed ¹. Additionally, [Juan et al., 2017] discuss tweaks and tricks as to how FFM was used in industry. Secondly, FFM is a classification based approach and treats all the unobserved feedback as equally negative. This drawback can be overcome by using learning-to-rank based approaches, which we discuss next.

¹<https://www.kaggle.com/c/outbrain-click-prediction/discussion/27892>

2.4 Collaborative Ranking

Most recommendations are presented in a sorted list with highest predicted score at the first positions. Recommendation can, therefore, be understood as a ranking problem. Therefore, learning-to-Rank for recommendations is a more realistic problem to solve, as compared to, the rating prediction problem addressed by standard CF approaches.

2.4.1 Learning-to-Rank

Learning-to-Rank (LTR) defines the task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance [Liu, 2009] for a given user. Motivated by automatically tuning the parameters involved in the combination of different scoring functions, LTR approaches were originally developed for Information Retrieval (IR) tasks and are grouped into three main categories: pointwise, listwise and pairwise. LTR models represent a rankable item – e.g. documents, offers etc. – given some context – e.g. a user – as a numerical vector.

In the context of RS, considering a set of users \mathcal{U} and a set of items \mathcal{I} , we aim to discover, for each user $u \in \mathcal{U}$ a total ordering over \mathcal{I} , where $i \succ_u i'$ implies that i is preferred to i' for u . Then, the goal is to learn a ranking function f , defined such that $f : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R}$ preserves the preference order as much as possible. That is, given a user u , for all $i \succ_u i'$, we want f to satisfy $f(u, i) \succ f(u, i')$. Over the past years, several ways to learn the ranking function f have been proposed, and they can be classified into three groups.

Pointwise approaches Pointwise approaches [Crammer and Singer, 2001; Li et al., 2007] assume that each item pair has an ordinal score. Ranking is then formulated as a regression problem, in which the rank value of each item is estimated as an absolute quantity. Formally, in point-wise approaches, the function f directly approximate $f(u, i) \approx r_{ui}, \forall (u, i) \in \mathcal{S}$. In this case, the ordered sequence of $f(u, i_1), \dots, f(u, i_m)$ is the total ordered list of preference for a user u .

In the case where relevance judgments are given as pairwise preferences (rather than relevance degrees), it is usually not straightforward to apply these algorithms for

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

learning. Moreover, pointwise techniques do not consider the interdependency among items, so that the position of items in the final ranked list is missing in the regression-like loss functions used for parameter tuning.

Listwise approaches On the other hand, listwise approaches [Shi et al., 2010; Xu and Li, 2007; Xu et al., 2008] take the entire ranked list of items for each query as a training instance. As a direct consequence, these approaches are able to differentiate documents from different queries, and consider their position in the output ranked list at the training stage. Listwise techniques aim to directly optimize a ranking measure, so they generally face a complex optimization problem dealing with non-convex, non-differentiable and discontinuous functions. Among popular approaches, we can cite CliMF [Shi et al., 2012], which optimizes a lower bound of the smoothed reciprocal rank of “relevant” items in ranked recommendation lists to learn a ranking function which operates on a binary rating matrix and uses a variant of latent factor collaborative filtering. [Shi et al., 2013] proposed an extension of CliMF that takes into account ratings with multiple level of relevance and optimizes a smooth approximations of the Expected Reciprocal Rank (ERR). Finally, CoFiRank [Weimer et al., 2007] uses a matrix factorization technique with a trace norm regularization on the factors) to handle explicit feedback by optimizing various losses including a smooth approximation of the Normalized Discounted Cumulative Gain (NDCG).

Pairwise approaches Finally, in pairwise approaches [Cohen et al., 1999; Freund et al., 2003; Joachims, 2002; Pessiot et al., 2007], the ranked list is decomposed into a set of item pairs. Ranking is therefore considered as the classification of pairs of items, such that a classifier is trained by minimizing the number of misorderings in ranking. Therefore, in this case, the ranking function $f(u, i)$ does not try to approximate r_{ui} , but rather focus on preserving the relative order of preferences between two ratings given by the same user.

In the test phase, the classifier assigns a positive or negative class label to an item pair that indicates which of the items in the pair should be ranked higher than the other

one. More formally, the goal is to minimize a risk function

$$\mathcal{L}(f) = \mathbb{E} \left[\frac{1}{|\mathcal{J}_u^+| |\mathcal{J}_u^-|} \sum_{i \in \mathcal{J}_u^+} \sum_{i' \in \mathcal{J}_u^-} \mathbb{1}_{y_{i,u,i'} f(i,u,i') < 0} \right], \quad (2.4.1)$$

where \mathcal{J}_u^+ and \mathcal{J}_u^- are the sets of preferred and non-preferred items, respectively, for a given user u ; $y_{i,u,i'} \in \{-1, +1\}$ is the desired output, and is defined over each triplet $(i, u, i') \in \mathcal{J}_u^+ \times \mathcal{U} \times \mathcal{J}_u^-$ as:

$$y_{i,u,i'} = \begin{cases} 1 & \text{if } i \succ_u i', \\ -1 & \text{otherwise.} \end{cases} \quad (2.4.2)$$

Typical pairwise losses considered in the case include the Hinge function, the exponential function or surrogate of the logistic loss [Chen et al., 2009].

Next, we present some popular pairwise ranking approaches that were successfully applied in the context of recommender system built to handle implicit feedback, in more details.

2.4.2 Pairwise-Ranking for Recommendation Systems

Bayesian Personalized Ranking (BPR) [Rendle et al., 2009] propose a Bayesian analysis of the pairwise ranking problem, implicitly assuming that users prefer items that they have already interacted with, at some other time. More precisely, given θ the set of parameters of a model (e.g. factorization matrix), BPR aims to maximize $p(\theta | \succ_u) \propto p(\succ_u | \theta) p(\theta)$ posterior probabilities. Following this formulation, the optimization of θ can be achieved through the optimization of criterion, namely BPR-Opt, which is related to the AUC (Area Under the Curve) (i.e., ROC curve) metric and optimizes it implicitly. Let us denote the optimization function of BPR-Opt by $F(\theta)$, $\text{BPR-Opt} \rightarrow F(\theta)$

The gradient of BPR-Opt with respect to the model parameters is, then, expressed as:

$$\nabla_{\theta} F = \sum_{u,i,i' \in \mathcal{S}} \frac{\partial}{\partial \theta} \ln \sigma(\hat{y}_{u,i,i'}) - \lambda \frac{\partial}{\partial \theta} \|\theta\|^2$$

where, θ are the model parameters, $\hat{y}_{u,i,i'}$ is the prediction that i is preferred over i' by

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

u (i.e. $f(u, i, i')$), σ is the Sigmoid function and \mathcal{S} is the training data

Algorithm 1 presents the procedure for learning the parameters in BPR, where one can use Stochastic Gradient Descent to optimize the BPR-Opt criterion.

Algorithm 1 BPR: Learning phase

Input : γ - learning rate ; λ - regularization parameter

Output : θ

Initialize θ

repeat

 Draw randomly (u, i, i') from \mathcal{S}

$\theta \leftarrow \theta + \gamma \left(\frac{e^{-\hat{y}_{u,i,i'}}}{1+e^{-\hat{y}_{u,i,i'}}} \cdot \frac{\partial}{\partial \theta} \hat{y}_{u,i,i'} + \lambda \cdot \theta \right)$

until convergence

Rank-ALS [Jahrer and Tscher, 2012] came up with the ranking based formulation of collaborating filtering for implicit feedback. The pairwise ranking objective function they minimize is the following:

$$\arg \min_{\mathbf{U}, \mathbf{V}} \sum_{u,i} c_{u,i} \sum_{j \in \mathcal{I}} s_j [(\mathbf{V}_i^\top \mathbf{U}_u - \mathbf{V}_j^\top \mathbf{U}_u) - (r_{ui} - r_{uj})]^2 \quad (2.4.3)$$

[Takács and Tikk, 2012], then used Alternating Least Squares for minimizing the objective function of 2.4.3 and coined the term RankALS for their algorithm. In the equation 2.4.3,

$c_{u,i}$ is the extent to which we penalize the error on user u and item i . Here, the authors assumed $c_{ui} = 0$ if $r_{ui} = 0$, and 1 otherwise. This setting selects user-item pairs corresponding to positive feedback. s_j sets the importance weight to be given to the j -th item in the objective function.

Hybrid approaches [Balakrishnan and Chopra, 2012] use Probabilistic Matrix Factorization (PMF) as first step. Then, they use pointwise and pairwise Learning to Rank methods (given in [Burges, 2010]) by using features learned during the first step of PMF. A very similar model is built by [Volkovs and Zemel, 2012], who also do PMF at the first step and use neighborhood approach for reducing the feature space.

[Liu and Aberer, 2014] optimize a pairwise Learning-to-rank loss, whereas [Lee et al., 2014] optimize a structured output loss. Finally, [Guillou, 2016], in his thesis worked on Ranking using (No-)Click Implicit Feedback in sequential recommendation of multiple items.

Pairwise Ranking with Neural Networks Perhaps the first Neural Network model for ranking is RankProp, originally proposed by [Caruana et al., 1995]. RankProp is a pointwise approach that alternates between two phases of learning the desired real outputs by minimizing a Mean Squared Error (MSE) objective, and a modification of the desired values themselves to reflect the current ranking given by the net. Later on [Burgess et al., 2005] proposed RankNet, a pairwise approach, that learns a preference function by minimizing a cross entropy cost over the pairs of relevant and irrelevant examples. SortNet proposed by [Rigutini et al., 2008, 2011] also learns a preference function by minimizing a ranking loss over the pairs of examples that are selected iteratively with the overall aim of maximizing the quality of the ranking. The three approaches above consider the problem of Learning-to-Rank for IR and without learning an embedding.

2.5 Deep Learning for Recommender Systems

Deep learning has proved its mettle in Speech Recognition, Computer Vision and Natural Language Processing and in recent years, there have been significant advances in deep learning applications for recommender systems.

For instance, deep learning has been used in collaborative filtering [Covington et al., 2016a; Dai et al., 2016; Elkahky et al., 2015; He and McAuley, 2015; Qu et al., 2016; Salakhutdinov et al., 2007; Wang et al., 2014, 2016; Wu et al., 2016; Zheng et al., 2016]. Recurrent Neural Networks (RNNs) being the model of choice for sequential type data, session-based recommendations have been done using RNNs [Chatzis et al., 2017; Hidasi and Karatzoglou, 2017; Hidasi et al., 2015, 2016; Quadrana et al., 2017; Ruocco et al., 2017; Smirnova and Vasile, 2017; Suglia et al., 2017; Tan et al., 2016; Twardowski, 2016] and in feature extraction directly from content [Bansal et al., 2016; He and McAuley, 2016; McAuley et al., 2015; van den Oord et al., 2013]. However, since our method in further chapters are based on embeddings and using learning to

rank in deep learning framework, this section is dedicated to discussing methods surrounding those ideas and approaches.

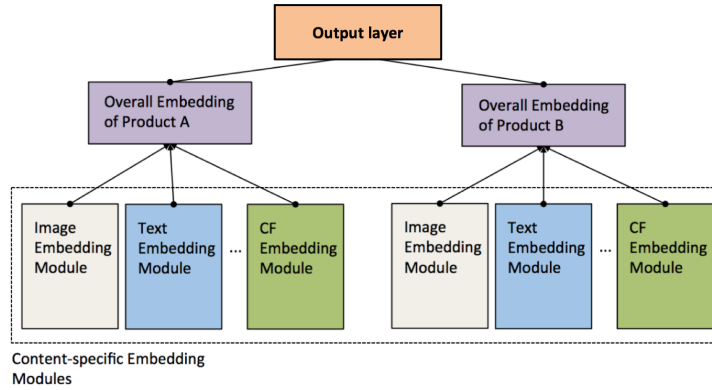


Figure 2.3 – Content2Vec architecture combines content specific modules to produce embedding vector for each product, then uses these vectors to compute similarities between products. Figure taken from [Nedelec et al., 2017]

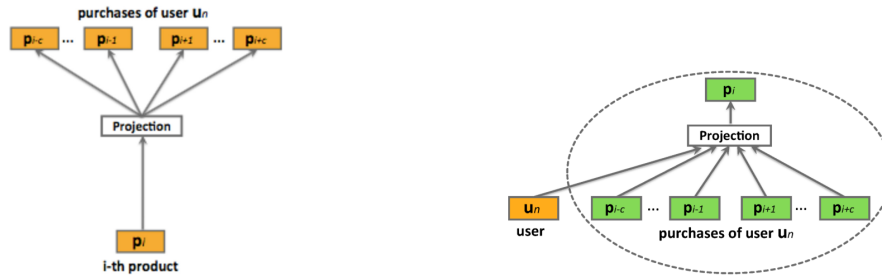


Figure 2.4 – Architecture of Prod2Vec and User-Prod2Vec [figure from [Grbovic et al., 2016]]

2.5.1 Representation Learning (RL) with Embeddings

Vector space model is well known in Information retrieval, in which documents are represented as vector. Idea of projecting words to a k -dimensional space has also started becoming popular in recent years. The major idea behind latent space projections and putting objects in a different and continuous dimensional space, is that the objects will have a representation (a vector) that has more interesting semantic characteristics than basic objects. There are various ways of representing words in vector

form. The most naive form of representing words is 1-hot encoding. Given a vocabulary of words with their positions in vocabulary fixed, 1-hot representation is a vector based representation of the word in which all the entries are zero except at the position of the word. But, the drawback of such a representation of words is due to the fact that they cannot model meaningful semantic relationships among words. Research in vector representations of words has taken off since the work of [Mikolov et al., 2013a], who represent words as embedding vectors. These models are based on a distributional hypothesis stating that words, occurring in the same context with the same frequency, are similar. In order to capture such similarities, these approaches propose to embed the word distribution into a low-dimensional continuous space using Neural Networks, leading to the development of several powerful and highly scalable language models such as the Word2Vec Skip-Gram (SG) model [Mikolov et al., 2013b,c; Shazeer et al., 2016]. Word2Vec models come in two flavours: Skip-gram model and Continuous bag of words (CBOW) and were first applied efficiently in Natural language processing tasks [Bengio et al., 2003; Mikolov et al., 2013a,d; Pennington et al., 2015]. Word2Vec maximizes the probability of the *context* given the target word. Neural language models and word embeddings, in particular, have proven themselves to be successful in many natural language processing tasks including speech recognition, information retrieval and sentiment analysis.

This idea of words occurring in a sequence paves the way for their application in RS also as items in RS are also consumed in sequence and prediction of context of items given a particular item sets an ideal analogy between word representations and item representations. The recent work of [Levy and Goldberg, 2014] has shown new opportunities to extend the word representation learning to characterize more complicated pieces of information. In fact, this paper established the equivalence between SG model with negative sampling, and implicitly factorizing a point-wise mutual information (PMI) matrix. Further, they demonstrated that word embedding can be applied to different types of data, provided that it is possible to design an appropriate context matrix for them. Next, we demonstrate how embeddings and vector representations of users and items can also be learned using neural networks and various works which have applied this idea to RS.

2.5.2 Users and Items Representation Learning (RL) with Embeddings

This idea has been successfully applied to recommendation systems where different approaches attempted to learn representations of items and users in an embedded space in order to meet the problem of recommendation more efficiently [Covington et al., 2016b; Grbovic et al., 2015; Guàrdia-Sebaoun et al., 2015; He et al., 2017; Liang et al., 2016]. In recommendations, notion of words is replaced with items in a session/user-profile. Various approaches have been developed on this idea, such as, Item2Vec [Barkan and Koenigstein, 2016b], Prod2Vec, Bagged-Prod2Vec and User-Prod2Vec [Grbovic et al., 2016], Meta-Prod2Vec [Vasile et al., 2016a] and Content2Vec [Nedelec et al., 2017].

In [He et al., 2017], the authors used a bag-of-words vector representation of items and users, from which the latent representations of latter are learned through word-2-vec. [Liang et al., 2016] proposed a model that relies on the intuitive idea that the pairs of items which are scored in the same way by different users are similar. The approach reduces to finding both the latent representations of users and items, with the traditional Matrix Factorization (MF) approach, and simultaneously learning item embeddings using a co-occurrence shifted positive PMI (SPPMI) matrix defined by items and their context. The latter is used as a regularization term in the traditional objective function of MF. Similarly, in [Grbovic et al., 2015], the authors proposed Prod2Vec, which embeds items using a Neural-Network language model applied to a time series of user purchases. This model was further extended in [Vasile et al., 2016b] who, by defining appropriate context matrices, proposed a new model called Meta-Prod2Vec. Their approach learns a representation for both items and side information available in the system. The embedding of additional information is further used to regularize the item embedding. Inspired by the concept of sequence of words; the approach proposed by [Guàrdia-Sebaoun et al., 2015] defined the consumption of items by users as trajectories. Then, the embedding of items is learned using the SG model and the users' embeddings are further used to predict the next item in the trajectory. In these approaches, the learning of item and user representations are employed to make prediction with predefined or fixed similarity functions (such as dot-products) in the embedded space.

2.6 Diversity in Recommender Systems

More recent research on recommender systems have started to focus on tackling the problem of *how* we recommend and not just *what* we recommend to the users. The ideal balance of how much diverse, relevant or novel the top recommended items are depends on the user in question. Although, the objective in recommendations is usually to have fewer flops on the top, inducing more diversity in the top items ensures that user may prefer to interact with at least some items in contrast to the situation where we introduce just monotonous relevant items. In recent works [Herlocker et al., 2004; McNee et al., 2006], it has been shown that only recommending relevant items (within the semantics of items which are the more likely to be of interest for users) has its limits and that adding notions of diversity and discovery in the process can highly increase the performance of a recommendation system [Bradley and Smyth, 2001; McSherry, 2002; Smyth and McClave, 2001; Zhang and Hurley, 2008].

In past years, approaches that propose to tackle the problem of finding an accuracy-diversity trade-off mainly rely on the re-ranking, a.k.a, Maximum Marginal Relevance principle, introduced originally in [Carbonell and Goldstein, 1998]. The re-ranking procedure is in two steps: first produce a list of the most relevant items for each user, using some individual scores $s(u, i)$, $\forall u \in \mathcal{U}$ and $i \in \mathcal{I}$; then re-rank the previously obtained list to enhance diversity using various diversity metrics [Deselaers et al., 2009; Drosou and Pitoura, 2009; Zhang and Hurley, 2008; Ziegler et al., 2005].

There have been works on strategies which do not involve re-ranking but clustering of items. For example, [Zhang and Hurley, 2009] partition the user's profile into cluster of items and recommend items from these clusters. In [Boim et al., 2011], authors cluster the items and then recommend a set of representative items, one for each cluster. [Li and Murata, 2012], use multi-dimensional clustering in order to provide diversified recommendations. [Shi, 2013] use graph-based approach and pose the problem as cost-flow to do bi-clustering and non-negative matrix factorization thus increasing the probability of non-tail items.

A recent article proposed to avoid re-ranking by directly optimizing a loss that takes into account both the diversity and the accuracy while building a list of items for each user. In (Learning to Recommend Accurate and Diverse Items [Cheng et al., 2017]), they consider the problem as a structural learning problem, where the set of

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

recommended items is optimized through structural SVM and a loss function combining diversity and accuracy. The main drawback of this approach is that, due to computational issue, they start by selecting a set of candidate items for each user, by only keeping items preferred by the user in the past.

There also have been works on multi-objective optimization, which try to optimize different objective functions for accuracy, diversity etc. For example, [Ribeiro et al., 2012] use the concept of pareto-efficiency to optimize accuracy, diversity and novelty simultaneously. [Su et al., 2013] include diversity term in matrix factorization objective function. [Hurley, 2013] incorporate diversity in learning to rank objective. [Wasilewski and Hurley, 2016b] also add diversity term in constrained Probabilistic Latent Semantic Analysis (PLSA).

In RankALS [Wasilewski and Hurley, 2016a], a diversity regularization term is added, thus taking into account diversity in a single step learning. The objective that they intend to minimize is given by

$$\mathcal{L}_{RankALS}(\Theta) + \lambda \text{reg}(\mathbf{U}, \mathbf{V}),$$

where λ is the parameter controlling the amount of diversity. The loss $\mathcal{L}_{RankALS}$ is the one defined in Equation 2.4.3. The authors derived various forms for the regularization term from the expected intra-list diversity (EILD) metric (which we define in section 2.7), all based on a distance matrix between items using some available characteristics (i.e. the genre of movies for the problem of movie recommendation).

Diverse and Novel Recommendations using Re-inforcement Learning

Reinforcement learning methods solve sequential decision-making tasks, where the decision is taken under uncertainty. At each time step, the system made of agent and environment is in a given state. The agent takes an action, and gets a reward or a cost of taking that action. In the end, the goal is to maximize the cumulative reward. One of the specific cases of Re-inforcement Learning is that of the point of view of the Multi-Armed Bandit setting, where there is only one state. Recommender Systems can be seen as Multi-Armed Bandit setting, where agent has to take action as to what to recommend next to the user so as to maximize the cumulative reward (clicks or the time-spent on the website). In a real-world recommender system, there is a

need for adaptability and reinforcement learning would embed perfectly to suit that need [Guillou, 2016]. One specific consequence of Re-inforcement Learning is that of *explore-exploit dilemma*. Exploit is the step, where we recommend an item which led to the best feedback in the past. Explore step enables us to recommend an item which hopefully brings information on the users tastes. [Tang et al., 2014; Xing et al., 2014; Zhao et al., 2013] have applied explore-exploit techniques to recommender systems. Indeed, in many recommendation applications such as news recommendations, it becomes important to adapt to ever-changing user interests and keep recommending new and diverse items to the user so that user doesn't get bored [Zheng et al., 2018].

2.7 Evaluation of Recommender Systems

Most researchers who suggest new recommendation algorithms also compare the performance of their new algorithm to a set of existing approaches. Such evaluations are typically performed by applying some evaluation metric that provides a ranking of the candidate algorithms (usually using numeric scores). RS are highly application oriented and they have specific goals and tasks. Evaluation should focus on the application goals and tasks [Gunawardana and Shani, 2015].

2.7.1 Prediction Accuracy

At the core of most RS systems lie the prediction model. Typical predictions consist of:

- What rating will a user give to an item ?
- Will the user select (e.g. click on) an item ?
- What is the order of usefulness of items to a user ?

Rating prediction accuracy

The RS is evaluated through its prediction for the items by users in test set, by comparing how close they are to the real ones. Most popular and widely used metrics are the following:

Mean Absolute Error (MAE) measures the average absolute deviation between the real and predicted rating.

$$MAE = \frac{1}{|J|} \sum_{(u,i) \in J} |\hat{r}_{ui} - r_{ui}|$$

Mean squared error (MSE) Compared to MAE, MSE puts the emphasis on large errors.

$$MSE = \frac{1}{|J|} \sum_{(u,i) \in J} (\hat{r}_{ui} - r_{ui})^2$$

The Root of the mean squared error(RMSE) is the square-root of the MSE value, and it is often employed in large number of collaborative filtering papers.

$$RMSE = \sqrt{MSE}$$

But, rating prediction task has been shown to have disadvantages. A system in which almost all ratings are around 3 in a 1 to 5 stars scale, would get a good evaluation score by predicting a 3 for every item. However, it would be more important to put more weight on high ratings to be able to correctly predict them from the point of view of the user. Moreover, high ratings do not necessarily mean high usage. People do not really watch more 5 star movies than the 3 star movies. The feedback given by the user has evolved from ratings to user-consumption of an item over time. Therefore, rating prediction has been deemed unfit from user's utility point of view[Basilico and Raimond, 2017].

Usage Prediction Accuracy

Usage prediction accuracy measures, as contrary to rating-prediction, evaluate and precise if the RS is capable of making relevant recommendations. They compare the list of recommended items by the RS with the ground truth of user's preferences. The relevancy of an item can be defined in different ways: in case of implicit feedback, such as clicks, an item may be considered relevant if the user clicks on it. In case of explicit feedback, such as ratings, an item may be considered relevant if the user provides

rating greater than 3.5 (on a scale of 1-5). Let $L(s)$ denote the recommendation list and R denote the relevant items for a user, the two metrics can be defined:

- the **Precision** measures the fraction of relevant items recommended in the list

$$Precision = \frac{|R \cap L(s)|}{|L(s)|}$$

- the **Recall** measures which fraction of the relevant items have been retrieved in the set of recommendation

$$Recall = \frac{|R \cap L(s)|}{|R|}$$

The scores of Precision and Recall can be conflicting and usually there is a trade-off between the two for any algorithm. Increasing the size of recommendation list will increase the recall, but decreases precision at the same time. This problem is often solved by using F_1 Score, which is harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

2.7.2 Ranking Measures


In RS systems, user usually receives a predicted sorted ranked list of recommendations containing top- k items. In an ideal case, ranked list of items should have highly preferred items higher in the list. Recommendations can, therefore, be studied as a ranking problem. Ranking measures have been used to evaluate how good is the evaluation by directly optimizing ranking measures.

Mean Average Precision@ k (MAP@ k) Precision@ k is defined as the precision (i.e. the percentage of relevant items among the first k recommendations) at the position k in the ranked results. Average Precision@ k (AP) is computed by taking the average of Precision@ i , $\forall i \in [1, k]$:

$$AP@k = \frac{1}{\# \text{ relevant at } k} \sum_{i=1}^k Precision@i \cdot rel(i),$$

where $rel(i)$ is an indicator function equalling 1 if the item at rank k is a relevant recommendation, zero otherwise. Then, the mean of $AP@k$ across all users is $MAP@k$. Below, we detail the step by step procedure of calculating $MAP@k$:

Precision@K

- Set a rank threshold K
- Compute % relevant in top K
- Ignores documents ranked lower than K
- Ex: 
 - Precision@3 : 2/3
 - Precision@5 : 3/5

Average of P@K

- Ex:  has $AvgPrec = \frac{1}{3}(\frac{1}{1} + \frac{2}{3} + \frac{3}{5}) \simeq 0.76$

The recommendation performance of all methods is evaluated on the test set. For each user in the test set, a ranking of items (only the items that the user interacted with) is generated and the mean average precision (MAP) is computed with a cut-off of different k . Then, the mean of these $AP@k$ (as defined in equation 2.7.2) across all relevant queries is the $MAP@k$. In the case of recommendations, $MAP@k$ is the $AP@k$ across multiple rankings of all users.

2.7.3 Diversity Measures

Most of the RS focus on the relevance of items being recommended to the user. By doing that, RS increase the sale of popular items. Also, these monotonous recommendations may also bore user and may mean less engagement with the system. In order to overcome these issues, RS often try to introduce diversity in their recommendations. Diversity of recommendations is often computed by computing Expected Intra-List Distance (EILD). Intra-List distance of any list $L(s)$ of items recommended to a particular user is given by:

$$ILD(L_u) = \frac{1}{N(N-1)} \sum_{i,j \in L(s)} d(i,j) \quad (2.7.1)$$

EILD is then given by averaging over all users:

$$EILD = \frac{1}{|U|} \sum_{u \in U} ILD \quad (2.7.2)$$

Distance $d(i, j)$ between two items i and j is computed using meta-data of items such as item-genre, item-category or item-embeddings. High value of EILD indicates high diversity. For more detailed reading on diversity metrics, one may refer to [Castells et al., 2015]

2.7.4 Online-Testing

In online-testing evaluation takes place within the real application on the real users. Typically, one or more recommender models are compared and user is assigned to one of the alternative systems uniformly at random, so that comparisons between alternative recommender systems is fair. Usually, it is also beneficial to single out different concepts. For example, if we are testing the accuracy of the system, user interface must be fixed across different recommender models. Many real-world systems employ online testing systems because of their advantages over offline counterparts.

Online-testing gives results over real users. Online-evaluation takes into account various factors which might effect real-time recommendations. There could be diverse factors effecting recommendations such as the user’s intent, the user’s personality, such as how much novelty or diversity vs. how much risk they are seeking, the user’s context, how much they trust the system and the interface through which the recommendations are presented. Online evaluation provides strongest evidence to the true value of the system by taking into account all the above mentioned factors. Performance is measured on the real application and results are trustworthy.

But, there are various problems also. Online-testing impacts real users and hence test system must be decent enough because varying user experience may be deemed bad. A test system that provides irrelevant recommendations, may discourage the test users from using the real system ever again. An extensive offline study must, therefore, be performed before doing online experimentation and an evidence should be obtained that candidate algorithms are reasonable enough to be tested online. Online-testing is also a lengthy process and may take long time. A more in-depth analysis on online

evaluation has been provided in [Gunawardana and Shani, 2015], who provide more details on significance testing and confidence intervals in online-evaluation.

A/B Testing A/B testing (bucket tests or split-run testing) is a randomized experiment with two variants, A and B. A/B tests are controlled experiments with thousands of users, which are applied to establish causal relationships between new treatment and change in user behavior with high probability [Kohavi, 2015]. Two versions (A and B) are compared, which are identical except for one variation that might affect a user’s behavior. Online A/B testing is generally used by companies to evaluate the impact of new technology by running it in a real production environment. Each new software implementation is tested by comparing its performance with the previous production version through randomized experiments. As an example, moving credit card offers from Amazon’s home page to shopping cart page boosted profits by tens of millions of dollars ¹. Good A/B metrics are of critical importance in order to make sound data-driven decisions. [Machmouchi and Buscher, 2016] do an in-depth study on the principles of design of online A/B metrics.

However, [Gilotte et al., 2018] have mentioned concerns for using online-version of A/B testing and design efficient offline estimators for offline A/B evaluation. [Joachims and Swaminathan, 2016] also do an indepth study on offline counterfactual evaluation of online A/B metrics

2.8 Conclusion

In this chapter, we described the state-of-the-art personalized recommendation techniques. We first discussed content based recommendation techniques. Then, we describe in detail collaborative filtering techniques being used. We go on to describe, in detail, learning-to-rank based methods for implicit feedback. We go on to detail the methods being used for representation learning for implicit feedback. Then, we describe the methods to make recommendations, not just relevant, but diverse. We finally discuss offline and online methods for evaluating recommendations.

¹Kohavi, Ron, and Stefan H. Thomke. ”The Surprising Power of Online Experiments: Getting the Most Out of A/B and Other Controlled Tests.” Harvard Business Review 95, no. 5 (September-October 2017): 7482.

2. RECOMMENDER SYSTEMS: STATE-OF-THE-ART AND EVALUATION

Chapter 3

Data-collections

3.1 Introduction

Nowadays, there are multiple learning based engines for optimizing the performance of advertising campaigns. Most of these engines are developed in the sense to be generic and adaptable for any type of advertiser on Internet, and allow to operate on different marketing axes, including commercial performance. A competitive engine has precise campaign objectives defined according to quantitative criteria, whether financial (profitability), media (traffic) or commercial (conversion, registration, purchase), and can achieve these objectives through a fine user targeting and sophisticated algorithms allowing to decide which ads should be displayed or when to stop presenting a given ad to the user. This fine ad targeting is primarily based on the collection and processing of the browsing history of the users, that can be traced using web cookies. Therefore, our first goal, in this thesis, is to collect, register and extract enough data in order to perform a first offline evaluation of the proposed models.

Hereafter, we describe two datasets, KASANDR and PANDOR that we extracted from Kelkoo's and Purch's traffic, respectively. They are designed to investigate a wide range of recommendation algorithms as they include many contextual features about both customers and proposed offers. For comprehensiveness, a description of side information and statistics are presented. The description of these datasets are published in SIGIR'17 [Sidana et al., 2017] and RecSys'18 [Sidana et al., 2018b].

3.2 Collection of the data

For building state of the art and novel RS models, the following constraints are taken into account. There are two components to the recommendation model: offline and online. Offline model is built on daily basis and takes long term user-personalizing into account, while the online model needs to be updated on hourly basis and takes context of the user as well as the model (built during the offline phase) into account. To make real time recommendations, various aggregate statistics, such as, number of unique users, number of returning users (within the month), number of new users (within the month), number of actions by user (min, max, avg) need also to be maintained. Finally, offers need to be recommended in real time with a time window of less than ten milliseconds. Keeping in view all these requirements, for building offline model,

3. DATA-COLLECTIONS

data is pushed into HDFS by batch importers at Kelkoo and Purch in compressed format (.parquet) each day. For doing online updates of the model, data is pushed by real time importers into AEROSPYKE based database. Kelkoo's one month logs are big enough (950 GB uncompressed) to not fit in one system. This leads to a lot of scalability issues, which come up when pre-processing data of this size. SPARK, which is a technology developed for handling big data and building machine learning models in a distributed manner, was used to do pre-processing and build dataset formats on which RS baselines can be built.

There were numerous bugs found in the initial stages of cleaning and pre-processing the data. We found that:

- We found that maximum number of the clicks were done by bots and not a human and many users have done no click at all.
- We found that many offers which were being clicked were never shown to the users.
- Users were tracked by maintaining cookies and this user-tracking system was not profound.

We removed/minimized the effects of these problems before starting to build RS baselines over the data.

KASANDR

The dataset records interactions of Kelkoo's customers between June, 1st 2016 and June, 30th 2016. It is designed to provide useful information in order to create and develop effective algorithms for the recommendation task. Kelkoo's traffic can be broadly classified according to 4 service types: (1) Ads, (2) Kelkoo's Website, (3) Kelkoo's Partners, (4) Kelkoo Feed System (KFS) which are summarized in Table 3.1. Kelkoo has collaboration with around 1000 partners (publishers/affiliates) on which users are advertised with offers. Various scenarios in which database at Kelkoo gets populated can be broadly classified into 4 different types:

- User visits Kelkoo's website and enters a search keyword. In this case, 1 PageView, 1 SearchView (with unique SearchId), N OfferViews (all having unique OfferViewId, where OfferViewId is the concatenation of searchId and offerId) are generated. If the user does a click, 1 ClickView (with unique ClickId) is generated.
- User browsing through Kelkoo's or partner's website is shown an ad (either a standard ad, or the user is retargeted, or on the basis of user's context, for example, the content of the page user is browsing). In this case also, 1 PageView, 1 SearchView (with unique SearchId - search keywords generated based on the ad content) and N OfferViews (1 per offer) are generated.
- User enters search keywords in Kelkoo's partner's website which does not cache offers. For each such search, a new Search_Id is generated and hence new OfferViewId is generated (as OfferViewId is concatenation of Search_Id and Offer_Id). In this case, there is no way to confirm that offer was displayed to the user.
- User enters search keywords in Kelkoo's partner's website on which offers are cached. In this case several users can see the same set of offers cached by the partner, hence, generating the same OfferViewId. In this case also, it can not be said for sure that the offer is displayed to the user.

Table 3.1 – Counts of the number of clicks done for each service type.

Type	Ads	Kelkoo site	Partners' Api	Kelkoo Feed System
Count	597,513	1,320,958	10,396,319	2,650,391

PANDOR

PANDOR is also designed to provide useful information in order to create and develop effective algorithms for online advertising. The dataset records the behavior of users of Tom's Hardware website¹ during one month; from 1st April 2018 to 30th April 2018. PANDOR gathers implicit feedback in the form of both impressions and clicks, given by users who have interacted with Purch online ads displayed on web articles.

¹<http://www.tomshardware.com>

3. DATA-COLLECTIONS

Information is collected when a user, browsing through Purch’s websites (e.g. Tom’s hardware), is shown an ad (either a standard ad, or on the basis of user’s context, for example, the content of the page that the user is browsing). In this context, 1 PageView and N OfferViews (N being the number of displayed offers) are generated. Then, if the user clicks on one of the offers that is shown to him, 1 ClickId is generated.

Next, we present the main characteristics of each of these datasets.

3.3 KASANDR Dataset

In the following, we describe the structure of KASANDR and present its basic statistics.

3.3.1 Structure of the data

The dataset is divided into four main databases that contain implicit feedback (offer views, clicks) of the users that have interacted with Kelkoo ads as well as a lot of contextual information (for full details, see Table 3.2). For privacy reasons, the UserID, name of the merchant and source were anonymized. In terms of contextual features, we have the following attributes:

- All four main files contain information about the geographic location of the user and the timestamp of each interaction. As mentioned previously, the data were collected across 20 countries and we provide the country code associated with each user.
- The click file contains the category of clicked products. There are more than 650 categories, provided by Kelkoo, organized hierarchically (according to two levels). We provide an XML file that describes this hierarchy and contains categories’ ID and label.
- The search table contains details about the users query: the string used to retrieve offers (QueryString), the list of filters apply to some of the queries to refine the search and a Boolean feature that indicates whether or not the query is filled by the user in the search box (isPrompt).

Table 3.2 – Description of free-available files. `train_set` and `test_set` have been created from Click and Offers for training recommender algorithms and further details are in next section.

File name	Format	Features
Page_View	csv	UserId, CountryCode, Timestamp, Url
Search	csv	SearchId, UserId, CountryCode, isPrompt, Timestamp, QueryString
Offers	csv	OfferId, OfferViewId, UserId, OfferRank, Merchant, price, Timestamp, CountryCode
Click	csv	ClickId, UserId, OfferId, OfferViewId, CountryCode, Category, Source, Timestamp, Keywords, OfferTitle
Product_Cat	xml	id and labels of product category presented as a tree
train_set	csv	UserId, OfferId, Service Type, ProductCategory, Country, Merchant, Feedback (1 or -1)
test_set	csv	UserId, OfferId, Service Type, ProductCategory, Country, Merchant, Feedback (1 or -1)

Finally, we also provide the train set and the test set used in the next section. All these files and additional details about the features can be found on-line¹.

3.3.2 Basic statistics

Table 3.3 and 3.4 report some basic descriptive statistics of the whole data. As outlined in these tables, we gather actions made by 123 million users over 56 million offers. In total, over the 3 billion offers displayed to those users, only 16 million were clicked resulting in the mega-sparsity of KASANDR.

Table 3.3 – Overall Dataset Statistics: 2016-06-01 to 2016-06-30.

# of users	# of unique offers	# of offers shown	# of clicks
123,529,420	56,667,919	3,210,050,267	16,107,227

Table 3.4 – Overall Dataset Aggregate Statistics.

Sparsity	99.9999997848%
Average # of Offers Shown to 1 user	26
Maximum # of clicks done by 1 user	3,722
Minimum # of clicks done by 1 user	0
Average # of clicks done by 1 user	0.13
Average # of clicks done by 1 user (if user did at least one click)	1.71

Figure 3.1(a) shows that the number of users fall sharply as the number of clicks rise, and most of the times either 3 or 6 offers are shown to the users. Figure 3.1(b)

¹<http://ama.liglab.fr/kasandr/>, <http://archive.ics.uci.edu/ml/datasets/KASANDR>

3. DATA-COLLECTIONS

depicts how the number of users and the number of clicks vary during the month. We can see that both numbers remain stable over the weeks. In addition, as previously mentioned, the data is collected across 20 countries and most of the clicks are generated by France and Italy, followed by Germany (see Figure 3.1(c)).

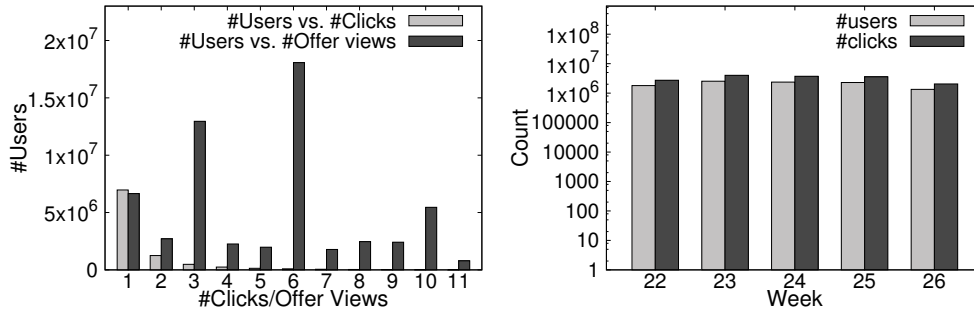


Figure 3.1 – (a) Number of clicks and number of offer views vs. number of users; (b) Number of clicks and number of users who did at least one click per week

From Table 3.5, one can observe that, over a month of data, very few number of users actually return to the system, when compared to the number of new users that emerge every week. This observation indicates that the time-window considered for making recommendation is important and gives information on how often a recommender model should be trained (offline) in order to provide relevant recommendations.

Table 3.5 – Number of new users and returning users per week.

Week Number	# New Users	# Returning Users
23	36,932,009	165,951
24	26,736,201	199,467
25	22,358,876	185,749
26	13,908,242	135,303

Kelkoo’s June data consists of many features which were made available to us to work with. For instance, it consists of nine different sources/partners from which data is collected. Apart from that, 461128 keywords and 680 different categories are contained in Click logs. Country-wise distribution of categories and merchants is shown in Figure 3.3

Next, we present PANDOR, another large scale dataset with rich text information.

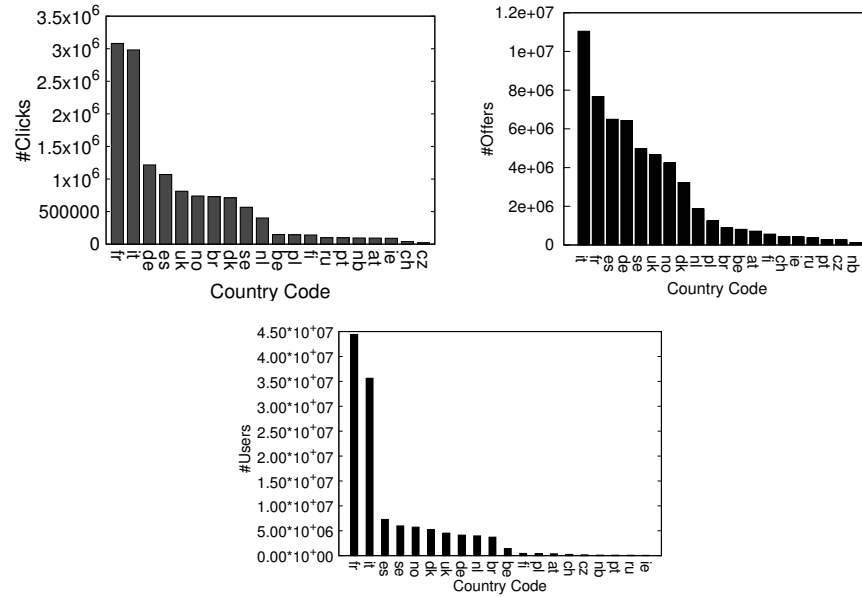


Figure 3.2 – (a) Number of clicks per country. (b) Number of offers per country. (c) Number of users per country.

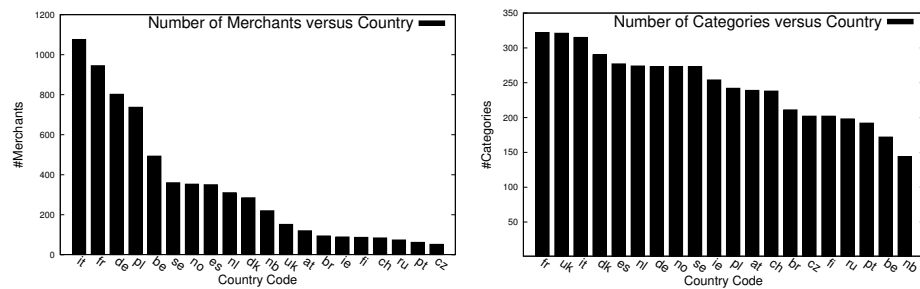


Figure 3.3 – (a) Number of Merchants per country. (b) Number of categories per country.

3.4 PANDOR Dataset

This Section presents in detail, another novel and publicly available dataset for online recommendation provided by Purch¹. The dataset, referred to as PANDOR, records the behavior of users of Tom’s Hardware website² during one month; from 1st April 2018 to 30th April 2018. PANDOR gathers implicit feedback in the form of both impressions and clicks, given by users who have interacted with Purch online ads displayed on web articles.

3.4.1 Structure of PANDOR

The dataset contains implicit feedback (offer views, clicks) of the users that have interacted with Purch’s ads (see Table 3.6 for details where we list the features we use to train our baselines). It should be noted that the dataset which we are going to make public also contains contextual information about offers such as keywords, titles, attributes and url of the page (and its anonymized text) on which offer was displayed. As some of the baselines we run later do not use contextual information, and to keep the comparison fair, we do not use them in the baselines we compare on PANDOR. However, baselines and our approach can be easily adapted to make use of all the contextual information we provide with this dataset. For privacy reasons, the UserID was anonymized. For each feedback (positive and negative), the Timestamp is recorded.

Table 3.6 – Description of train_set, test_set and Ratings files in PANDOR.

File name	Format	Features
Ratings	csv	utcDate, userId, offerViewId, offerId, wasClicked
train_set	csv	UserId, OfferId, Feedback (1 or -1), Timestamp
test_set	csv	UserId, OfferId, Feedback (1 or -1), Timestamp

Finally, we also provide the train set and the test set used in the next section. All these files and additional details about the features can be found online.³

¹<http://www.purch.com/>

²<http://www.tomshardware.com>

³For research purpose we will make available all the files along with all the contextual information as well as the codes that we used in our experiments and the pre-processed data sets.

3.4.2 Features of PANDOR

Some statistics are provided in Tables 3.7 and 3.8, highlighting the complexity of the proposed data, both in terms of sparsity and size. As outlined in Table 3.7, the datasets gather the actions of close to 2M users over 3.7K products. We further describe more contextual information present in PANDOR in Table 3.10. By one event, we mean the act of being shown a banner of advertisements to a user. As can be seen from that Table, PANDOR is also suitable for text based Recommender Systems. This text based information present in PANDOR makes it fit for applying topic modeling techniques for getting meaning semantics as a preprocessing step. Among the 48M interactions observed, only 337K resulted in a positive feedback, i.e., click. Furthermore, one can observe that the maximum number of clicks done by one user is 119 while the average number of clicks is below 0.057 (see Table 3.8).

Table 3.7 – Overall Dataset Statistics, from 2018-04-01 to 2018-04-30.

# of users	# of unique offers	# of offers shown	# of clicks
5,894,431	14,716	48,754,927	337,511

Table 3.8 – Overall Dataset Aggregate Statistics.

Maximum # of offers shown to 1 user	2,029
Average # of Offers Shown to 1 user	8.271
Maximum # of clicks done by 1 user	119
Minimum # of clicks done by 1 user	0
Average # of clicks done by 1 user	0.057
Average # of clicks done by 1 user (if user did at least one click)	1.351

Table 3.9 – Number of new users and returning users per week.

Week Number	# New Users	# Returning Users
14	1,387,876	99
15	1,298,355	592
16	1,447,586	647
17	1,395,424	774

From Figure 3.4(a), one can observe that the number of users fall sharply as the number of clicks rise. In addition, the majority of users were shown one offer (i.e.

3. DATA-COLLECTIONS

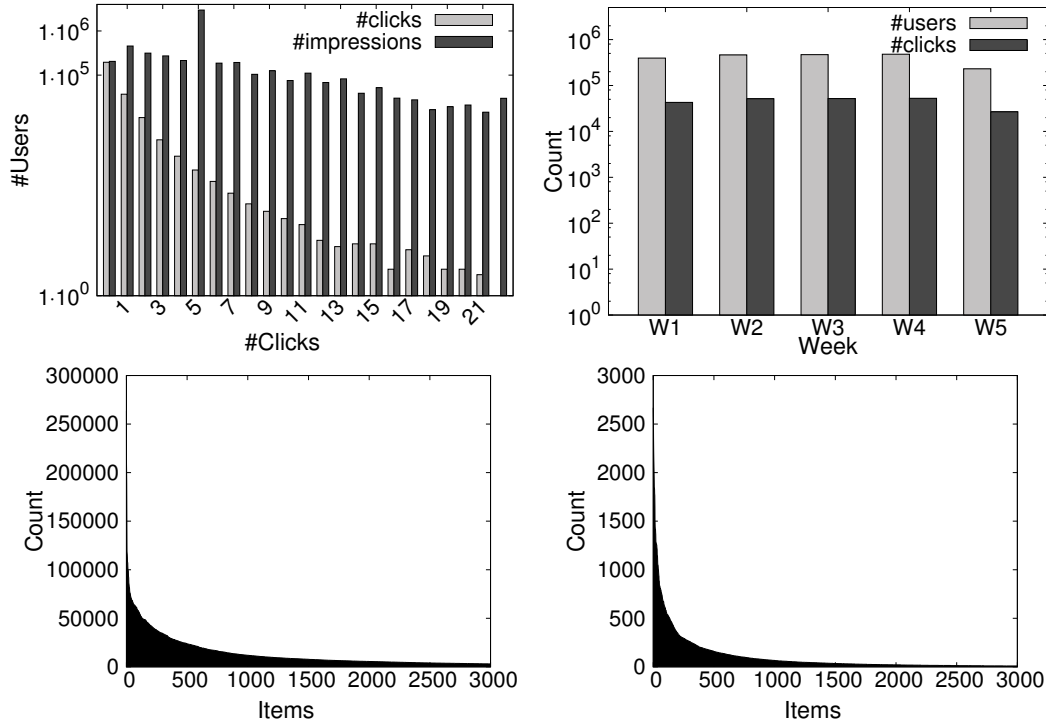


Figure 3.4 – (a) Number of clicks and number of offer views vs. number of users; (b) Number of clicks and number of users who did at least one click per week; (c) Long tail item : number of time each item is recommended; (d) Long tail item : number of time each item is clicked

Table 3.10 – Overall Dataset Textual Statistics, from 2018-04-01 to 2018-04-30.

# Events	48,602,664
# Events where user did at least 1 click	4,544,848
# Events which have at least 1 page text words	1,212,170
# Events which have at least 1 product text words	450,050
# Events which have at least 1 keyword	4,492,544
Page text vocabulary size	9,111
Product text vocabulary size	6,016
Keyword vocabulary size	543
# Offers which have at least 1 text word	2,701 (27.4%)
# Pages which have at least 1 text word	1,990 (28.1%)

impressions), while, the number of users that were shown 2 to 7 offers are quite balanced. Figure 3.4(b) depicts how the number of users and the number of clicks vary

during the month the dataset was collected. We can see that both numbers remain stable over the weeks. Finally, an important specificity of the dataset is that, at the time it was extracted, the actual recommendation system in production was mainly based on the popularity of the items, meaning that the ads displayed to any particular user were mostly related to the most clicked or sold products. Another part of recommendations system is based on LDA-based user profile similarity. As a result, the coverage of items is extremely low and the dataset presents what is referred to as the long-tail phenomenon or the popularity bias in the literature [Anderson, 2006; Park and Tuzhilin, 2008](see Figure 3.4(c)).

Other dataset collections We briefly mention the other datasets, we are going to use in experiments section 6.3 here. These collections are not the contributions of this thesis, but some popular collections, which RS community tend to use for benchmarking their models. In particular, we use ML-100K, ML-1M and NETFLIX. All of these are explicit feedback datasets and synthetically made implicit by considering rating \geq four as one (and less than four as 0). More details about these datasets and analysis will be done in Section 6.3

3.4.3 Summary

In this chapter, we presented novel datasets in order to encourage future research on recommendation systems using implicit feedback. They are designed to investigate a wide range of recommendation algorithms as it includes many contextual features about both customers and proposed offers. For comprehensiveness, a description of side information and statistics are presented.

Another interesting perspective include the integration of textual information available in KASANDR and PANDOR using the URL to retrieve the content of the page on which the item is presented, the tag associated to it, or the query string entered by the user for his search. For this purpose, models based on text mining, semantic analysis or natural language processing can be investigated. We also left aside other features in the experimentation such as the consumer's behavior w.r.t. the type of device that s/he is using or the price of the items which we believe that they can greatly impact the performance of RS.

Chapter 4

Extracting latent topics over timely related articles

4.1 Introduction

Recommendations are actions at a moment in time. This moment can be controlled by the user's actions such as visit time of the user or session length of the user. Time is a critical aspect in any RS [Basilico and Raimond, 2017]. Hence, some works have tried to address this aspect of time [Ding and Li, 2005; Koren, 2010; Rendle et al., 2010; Shani et al., 2005; Zimdars et al., 2013]. Recurrent Neural Networks (RNNs) have been used for session-based recommendations [Chatzis et al., 2017; Hidasi and Karatzoglou, 2017; Hidasi et al., 2015, 2016; Quadrana et al., 2017; Ruocco et al., 2017; Smirnova and Vasile, 2017; Tan et al., 2016; Twardowski, 2016]. Comprehensive survey has been done in time-aware recommendations [Campos et al., 2014].

On a similar note, textual data plays an important role in content-based recommendations. [Musto et al., 2016, 2017] use textual data in order to provide content-based recommendations. RS data are usually sparse with users providing very little feedback about their preferences. There are wide variety of textual data, which can provide an extra feedback signal. Textual data such as product-reviews gives an explicit preference signal of the user. On the other hand, page browsing, search-text, offer (title, tags and categories), which user clicked on, provide implicit preference of users. All these explicit or implicit feedback, if modeled properly, can help boost performance of RS models.

Topic modeling is a good way to exploit textual data and to extract relevant information and topics. These relevant topics can be used as a contextual information in wide variety of RS. In other words, the topics inferred from text can be used as supplementary feature in RS models.

This contextual information of topics can also be used in content-filtering based RS. For example, on a news website, a particular user has viewed/engaged with a particular set of page views. The text contained in these page-views can then be used to model the topics of interest to the user. These topics can then be used to make future recommendations to the user by building long term user-profile by aggregating the topics of pages which user has engaged in the past. The topics of interest of the user also evolves with time. This evolution of intent of the user with time is also of interest in making recommendations to the user.

All the above approaches rely on how well are we able to model the topics of

underlying document collections as well as on how these underlying topics and the intent of the user over these topics evolve over time. In this vein, we make use of both text and time and use topic-model based approaches.

In what follows, we present various topic-modelling techniques. In Section 4.3, we first define general-purpose topic modelling techniques. Then we present a novel temporal topic model, which we developed during the course of this thesis, in Section 4.2.2. We then depict the efficacy of the two novel temporal topic models on health-based recommendations by using *perplexity* in the results Section 4.5 of this chapter. Then, in experiments Section 6.4, we use topics derived from one of these temporal topic models as contextual information in Factorization Machines, a popular approach for RS with implicit feedback. This chapter is the extension of work published in SIGIR'16 [Sidana et al., 2016] and IEEE TKDE'18 [Sidana et al., 2018a]

4.2 General-purpose topic modelling

In this section, we describe two topic models, which are central to our work. These topic models are used to model general purpose topics. In Section 4.2.1, we describe, in detail, Latent Dirichlet Allocation (LDA). Then, in Section 4.2.2, we proceed with the description of the Topic Aspect Model (TAM), which, not only takes into account, the topics associated with document, but also aspects. Aspect, which we describe in next section, is underlying theme or perspective of the text contained in the document.

4.2.1 Latent Dirichlet Allocation (LDA)

LDA represents each document as a probability distribution over k topics [Blei et al., 2003]. Each topic z in turn is represented as a probability distribution ϕ_z over a set of words and both follow multinomial distribution. Figure 4.1, shows the plate diagram of LDA. The topic distribution of the document d is denoted by θ_d and the word distribution of the topic z is denoted by ϕ_z . In other words, topics assigned to words in document d follow a multinomial distribution with parameter θ_d and words are generated (once the topic z has been assigned) by again using a multinomial distribution with parameter ϕ_z . In this manner, each document holds its own multinomial topic distribution vector and each topic, in turn, holds its own multinomial word distribution

vector. The topic distribution θ_d of a document d and the word distribution ϕ_z of a topic z are, themselves, generated according to a Dirichlet (prior) distribution. In Figure 4.1, α is the dirichlet prior to the topic distribution θ and β is the dirichlet prior to word distribution ϕ . The prior distribution is our belief about topic and word distributions before seeing any data. We take Dirichlet as the prior, because, Dirichlet is the conjugate prior to Multinomial and this ensures that posterior distribution of topics and words is of the same type as their prior distribution. The vectorial parameters α and β of these Dirichlet distributions are assumed to be common to the whole corpus. The lesser the sum of α , more spread out is the weight to all the topics in a given document. If we consider M as the number of documents each of length and N_i as the number of words in the vocabulary, then generative process of LDA is as follows:

Generative Process of LDA

```

Choose  $\theta_i \sim Dir(\alpha)$ ;
Choose  $\phi_z \sim Dir(\beta)$  ;
for each of the word positions  $i,j$ , where  $i \in \{1, \dots, M\}$  , and  $j \in \{1, \dots, N_i\}$ 
do
    Choose a topic  $z_{i,j} \sim Mult(\theta_i)$ . ;
    Choose a word  $w_{i,j} \sim Mult(\phi_{k_{i,j}})$ .;

```

Values of α and β are often pre-chosen before running LDA. Lower values of Dirichlet parameters tend to make it more spread out, while large values tend to make the distribution more peaky. Hence, if we think the documents are about many topics, it may make more sense to take lower values of α parameter. Given the words of the document, the goal of the LDA, is to infer the topic assignments z to each of the words. Once the topic assignments to words have been inferred, topic distribution θ can be calculated. Exact inference of the posterior distribution of the latent variables z is intractable. In practice, a Markov chain Monte Carlo algorithm, namely, Gibbs Sampling [Andrieu et al., 2003] is used to do approximate inference of latent variables. In a Gibbs Sampler, new values of all the latent variables are iteratively sampled for each token i from the posterior probability conditioned on the previous state of the model (i.e., the current values for all other tokens). One can refer to [Blei et al., 2003] for more details on all the inference equations involved in LDA.

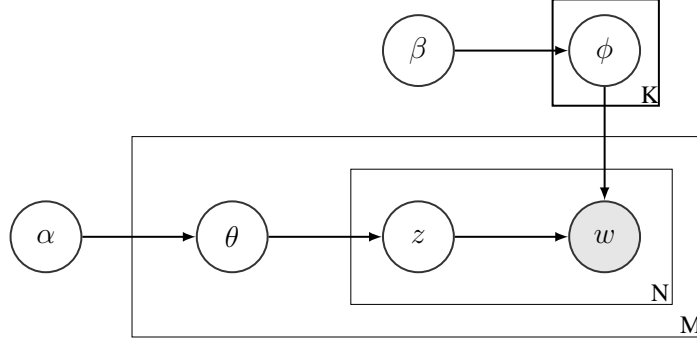


Figure 4.1 – Latent Dirichlet Allocation.

4.2.2 Topic-Aspect Model (TAM)

Paul and Girju [2010] came up with a topic-aspect model TAM. We show the plate diagram of TAM in Figure 4.2. In TAM also, each document is a probabilistic multinomial distribution over topics, denoted by θ in Figure 4.2. Each topic is a probabilistic distribution over words, denoted by ϕ . θ has a Dirichlet prior, namely α and ϕ has a Dirichlet prior β . Words within each topic are semantically related somehow.

The novelty of TAM compared to other topic models, is a second mixture component that can affect the nature of a documents content. We broadly define an *aspect* of a document as a characteristic that spans the document such as an underlying theme or perspective. In Figure 4.2, *aspect* is denoted as the variable y . Binary switching variable x determines if the word comes from the aspect-neutral word distribution or aspect-dependent distribution. A computational linguistics paper may have both a computational aspect and a linguistic aspect. For instance, the computational aspect of the SPEECH RECOGNITION topic might focus on Markov models and error detection, while the linguistic aspect might focus on prosody. Other computational linguistics topics would likewise have words that are characteristic of each aspect. x is drawn from a binomial distribution parameterized by π , which has a beta prior γ .

TAM also includes a additional mixture component to distinguish common words and functional words from topical words. All the common *background* words that appear independently of a document’s topical content are included in the case of $l = 0$. A common word like *using* would likely belong to background level, as it is not particularly topical. In the lower level $l = 1$, each word is associated with a topic. l is drawn from binomial distribution parameterized by the parameter λ .

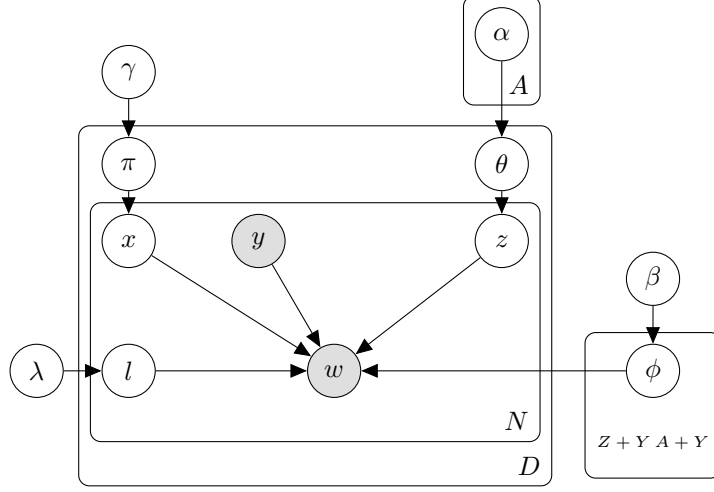


Figure 4.2 – Topic Aspect Model.

If the word is generated from the background model, the word is sampled from $P(word|l = 0, x = 0)$ or $P(word|l = 0, x = 1, aspect)$ depending on if the aspect-independent or -dependent model is used. If the word is generated from topical model, it is sampled from $P(word|l = 1, x = 0, topic)$ or $P(word|l = 1, x = 1, aspect, topic)$.

Just like the LDA model, Gibbs sampling can be used for inference and parameter estimation for TAM as well. In what follows, we propose the time-aware topic models as topics which are time-dependent tend to be more meaningful than time-oblivious topics in applications such as RS.

4.3 Temporal Latent Topic Models

In this Section, we describe topic models, which take time at which topics are inferred into account. First, we describe a simple yet effective technique which builds on the top of LDA in Section 4.3.1. Then, in Section 4.3.2, we describe a novel topic model, which treats time as an observed random variable inside TAM .

4.3.1 Temporal-LDA (TM-LDA)

In order to take into account the evolution of the underlying topics of a dynamic collection of documents with time (e.g., a microblog or a facebook page), Wang et al. (2012)

4. EXTRACTING LATENT TOPICS OVER TIMELY RELATED ARTICLES

introduced a modified version of the LDA model, TM-LDA [Wang et al., 2012]. In [Wang et al., 2012], TM-LDA was introduced to extend LDA with modeling evolution of topics of dynamic collection of documents over time. Topic distribution of the i -th document, θ_i is assumed to depend linearly on the topic distribution of the previous document, θ_{i-1} . At the heart of the algorithm lies the following equation.

$$\theta_i \approx \frac{\theta_{i-1} \cdot M}{\|\theta_{i-1} \cdot M\|_{\ell_1}} \quad (4.3.1)$$

where M is a $k \times k$ matrix, called the transition matrix, and k is the number of topics. To obtain the transition matrix, the authors propose to solve the following least squares problem ($\|\cdot\|_F$ denotes the Frobenius norm and X denotes the search space):

$$M = \arg \min_X \|A \cdot X - B\|_F \quad (4.3.2)$$

where A and B are as specified below.

$$A = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{i-1} \end{pmatrix}, B = \begin{pmatrix} \theta_2 \\ \vdots \\ \theta_i \end{pmatrix} \quad (4.3.3)$$

TM-LDA is quite elegant in modeling general purpose topics over time. But, one significant disadvantage of using TM-LDA, is the huge amount of postprocessing, which is required to model transition matrices.

4.3.2 Time-Aware Topic-Aspect Model

TAM can then be extended to include a document level characteristic a such as the one shown in Plate diagram 4.3. Document level characteristic could be anything such as *overall sentiment* of the document or *a disease* in case of health document. Indeed, recommender system documents are often accompanied by document reviews such as Amazon product reviews¹ and have an overall sentiment attached to them. a is drawn from multinomial distribution η with a Dirichlet prior σ . If a is considered as sentiment, then, possible values of a can be positive, negative or more generally, could

¹<http://jmcauley.ucsd.edu/data/amazon/links.html>

contain more values such as very-positive, positive, neutral, negative, very-negative.

Sentiment towards topics also evolves over time. For example, in news recommender systems, taste of users towards various topics keep changing with time and it becomes important that any RS we build to recommend news to users, is time-aware of such evolving interests. Hence, we introduce a random variable t for time in TAM as shown in Plate diagram 4.4. Here document level characteristic a such as sentiment is drawn depending on time t . Time t itself is drawn from a multinomial distribution ψ drawn from Dirichlet prior μ .

Generative process of time-aware TAM with document level characteristic a is as follows:

Generative process of time-aware TAM with document level characteristic a

```

Set the background switching binomial  $\lambda$ 
Draw a sentiment distribution  $\eta \sim Dir(\sigma)$ 
Draw A multinomials  $\psi_A \sim Dir(\mu)$ 
Draw word multinomials  $\phi \sim Dir(\beta)$  for the topic, sentiment, and background
distributions
for each message  $1 \leq m \leq D$  do
  Draw a switching distribution  $\pi \sim Beta(\gamma_0, \gamma_1)$ 
  Draw a sentiment  $a \sim Mult(\eta)$ 
  Draw a time stamp  $t \sim Mult(\psi_a)$ 
  Draw a topic distribution  $\theta \sim Dir(\alpha_a)$ 
  for each word  $w_i \in N_m$  do
    Draw aspect  $y_i \in \{0, 1, 2\}$ (observed)
    Draw background switcher  $l \in \{0, 1\} \sim Bi(\lambda)$ 
    if  $l == 0$ : then
      Draw  $w_i \sim Mult(\phi_{B,y})(a \text{ background})$ 
    else
      Draw  $x_i \in \{0, 1\} \sim Bi(\pi)$ 
      if  $x_i == 0$  : (Draw word from topic  $z$ ) then
        Draw topic  $z_i \sim Mult(\theta)$ 
        Draw  $w_i \sim Mult(\phi_z)$ 
      else
        (draw word from sentiment  $a$  aspect  $y$ )
        Draw  $w_i \sim Mult(\phi_{a,y})$ 

```

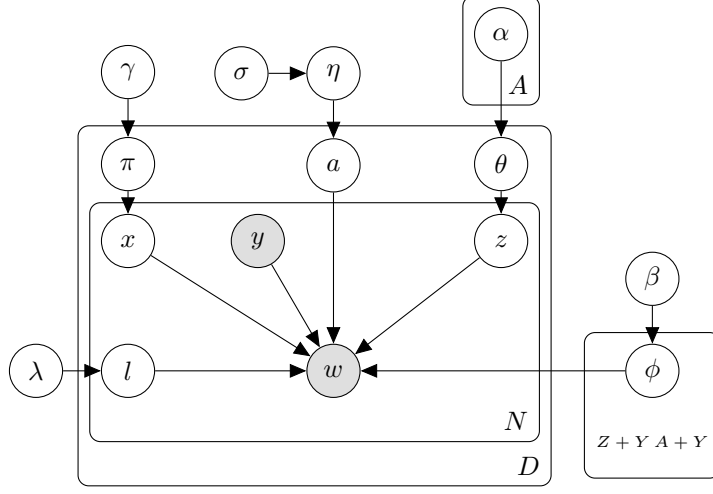


Figure 4.3 – Topic Aspect Model with a document level charateristic a . This a could be seen as over all sentiment of the document.

Document-level Gibbs sampling for a is given by following equation:

$$\begin{aligned}
 P(a_m | \mathbf{a}_{-m}, \mathbf{w}, \mathbf{t}, \mathbf{y}, \mathbf{x}, l) \\
 \propto P(a_m | \mathbf{a}_{-m}) P(t_m | \mathbf{t}_{-m}, \mathbf{a}, \mu) \\
 \prod_n^{N_m} p(w_{m,n} | \mathbf{a}, \mathbf{w}_{-(m,n)}, \mathbf{y}, \mathbf{x}, l)
 \end{aligned} \tag{4.3.4}$$

The inference equations for other latent variables stay the same as TAM and are detailed in [Paul and Girju, 2010]. In the next section, we divide data temporally into train and test and compare all the topic models on health text (tweets) in their ability to predict topics of non-seen future text. We use *perplexity* (which we describe in the next section) in order to do so.

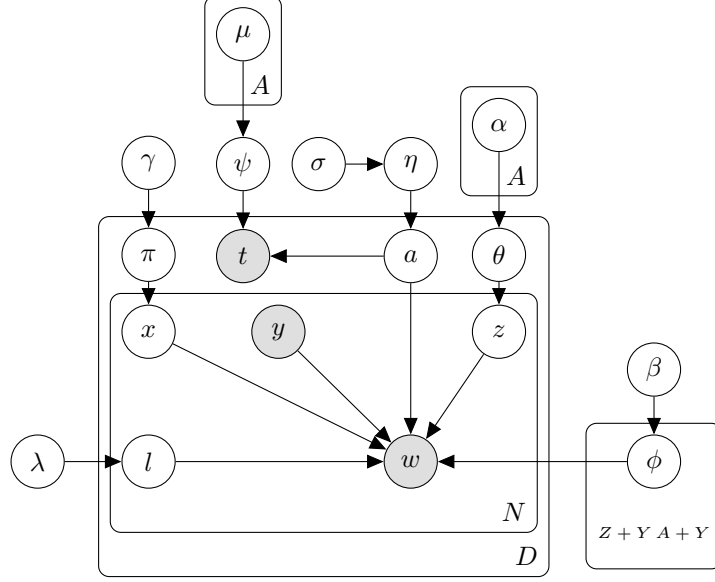


Figure 4.4 – Time-Aware Topic Aspect Model. Sentiment a is time-aware.

Table 4.1 – Mapping tweets to documents

Term	Description
\mathcal{P}	posts
\mathcal{G}	regions
\mathcal{T}	time periods
\mathcal{P}_g^t	posts from region g during time t
D_g^t	document-set built by mapping the content of each post $p \in \mathcal{P}_g^t$ to a document

4.4 Application to health monitoring on social media over time

Henceforth, we apply the topic models which we described so far with an application to health monitoring on Twitter over time. Twitter has become a major source of data for *early monitoring and prediction* in areas such as health [Manikonda and Choudhury, 2017], disaster management [Chowdhury et al., 2013] and politics [Davidson et al., 2017]. In the health domain, the ability to model transitions for ailments and detect statements like “people talk about smoking and cigarettes before talking about respiratory problems”, or “people talk about headaches and stomach ache in any or-

der”, benefits syndromic surveillance and helps measure behavioral risk factors and trigger public health campaigns. Notations we used in this section are summarized in Table 4.1.

4.4.1 TM-LDA applied to health documents

First model is a direct application of TM-LDA which is described in Section 4.3.1 to health documents and the resulting model is coined the term TM-ATAM [Sidana et al., 2016]. TM-ATAM, at its heart, solves following equation:

$$A_g^t \approx A_g^{t-1} \cdot M \quad (4.4.1)$$

where

$$A_g^{t-1} = \begin{pmatrix} \Theta_g^1 \\ \vdots \\ \Theta_g^{t-1} \end{pmatrix}, A_g^t = \begin{pmatrix} \Theta_g^2 \\ \vdots \\ \Theta_g^t \end{pmatrix} \quad (4.4.2)$$

M is an unknown transition matrix which is obtained by solving the following least square problem:

$$\min_M \|A_g^t - A_g^{t-1} \cdot M\|_F$$

TM-ATAM thus learns a transition matrix which is used to model health topics.

4.4.2 T-ATAM

T-ATAM is a direct application of time-aware topic aspect model, which was described in the Section 4.2.2 to health documents with the only difference that instead of sampling a sentiment for each document, we sample an ailment/disease for a document since the corpus we are using is health tweets [Sidana et al., 2018a]. In Figure 4.4, Document-level random variable a is treated as ailment. The plate diagram and generative process stays the same. For aspects, we use symptom, treatment and general related aspects in health documents.

Table 4.2 – Dataset Statistics

collection period (days)	235
#tweets	1,360,705,803
#tweets (health-related)	698,212
#tweets (health-related+geolocated)	569,408

4.5 Results

We conduct experiments to evaluate the performance of TM-ATAM and T-ATAM on the real world data. In section 4.5.1, we describe the data we use for experiments. Then, in section 4.5.2, we compare different topic models which we described above using perplexity.

4.5.1 Data

We employ Twitters Streaming API to collect tweets between 2014-Oct-8 and 2015-May-31. We use the *Decahose Stream*¹ which gives a 10% random sample of the total tweets generated each day. The collected tweets were subjected to two pre-processing steps. We removed retweets and tweets containing URLs; they were almost always false positives (e.g., news articles about the flu, rather than messages about a users health). Since our interest lies in public health discourse on social media, we only keep tweets containing one of 20,000 health-related keywords obtained from wrong-diagnosis.com. This website lists detailed information about ailments, symptoms and treatments. Resulting tweets were given to an SVM classifier [Cortes and Vapnik, 1995] with linear kernel and uni-gram, bi-gram and tri-gram word features. To train the classifier, a modest-sized sample of the original corpus was annotated through crowd-sourcing efforts where annotators were asked to label 5, 128 tweets. The precision and recall of the employed classifier are 0.85 and 0.44. In our case, we focused on high precision as high quality health tweets is a pre-requisite for both TMATAM and TATAM to function efficiently. Table 4.2 shows that out of the 1.36B tweets we collected, 698K were health-related.

¹<https://dev.Twitter.com/streaming/overview>

4.5.2 Comparison between models

The probabilistic *Ailment Topic Aspect Model* was designed specifically to uncover latent health-related topics in a collection of tweets [Paul and Dredze, 2011]. In this section, we compare the performance of TM-ATAM and T-ATAM against ATAM, TM-LDA and LDA.

Perplexity We use *perplexity*, an empirical measure often used in NLP.¹ Perplexity of a language model measures how accurately the model can explain previously unseen data/documents. Given a language model l and a document d , perplexity is defined as below.

$$Perplexity(l) = 2^{-\sum_{w_i \in d} \log p_l(w_i)} \quad (4.5.1)$$

This formula of perplexity for a document d can be converted to a formula of perplexity for a set of documents D_g^t as follows:

$$Perplexity_{D_g^t}(l) = 2^{-\sum_{w_i \in d} \log \frac{\sum_{d \in D_g^t} p_l(w_i)}{|D_g^t|}} \quad (4.5.2)$$

It denotes the perplexity of language model l on a document-set at geo-granularity g and temporal granularity t . Higher probability of words that occur in unseen documents results in lower perplexity and is hence better.

Figure 4.5 shows that TM-ATAM and T-ATAM consistently beats TM-LDA and ATAM in predicting future health topics on the test month by computing lower perplexity on the words of the tweets of the test month in all social media active states.

4.6 Conclusion

In this Chapter, we first describe how topics derived from topic models can be used for recommendations. To this end, we first described numerous topic models which can be used to model general topics of given document collections. In particular, we describe LDA, TAM and then introduce a document-level characteristic a as a random latent variable within TAM. This a is described as an overall sentiment of the document or

¹<https://en.wikipedia.org/wiki/Perplexity>

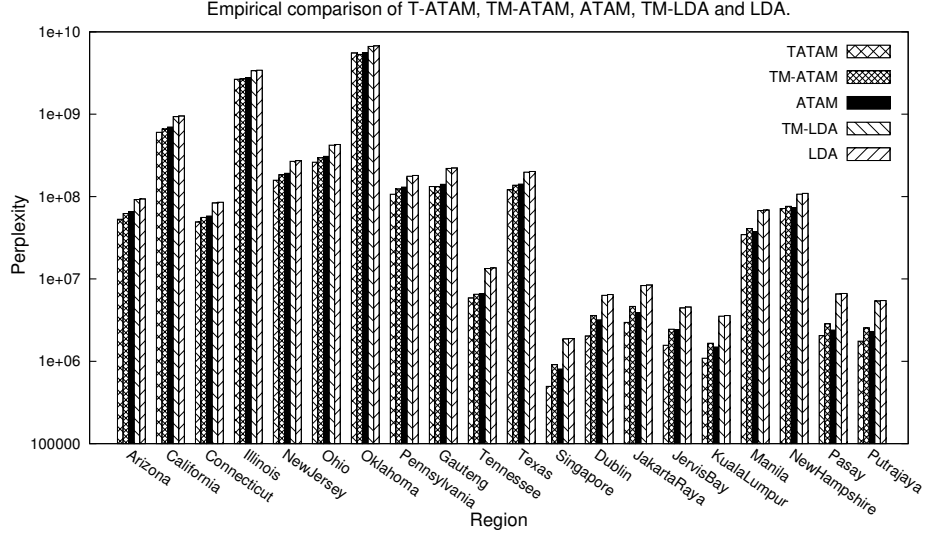


Figure 4.5 – Perplexity comparison of T-ATAM, TM-ATAM, TM-LDA and ATAM for top 20 social media active regions.

an ailment in case of health documents. Then, we introduce temporal nature of these topics and go on to describe time based topic models.

We first describe a simple yet effective extension of LDA for modelling topic transitions by solving a least squares problem between historical and present topic distributions. The topics, thus modeled, can be used as contextual information and fed into existing RS models. This model can be useful in the cases where, besides clicks, we just have item text (title, tag, description) information and can help in dealing with sparsity of data.

Then, we introduce time as a random variable in TAM. This time-aware TAM can be used to model topics of news articles and to build profiles of the user by aggregating the topics of the news articles which user has engaged in. These profiles can then be used to recommend future articles to the user.

Finally, we show the effectiveness of time-based topic models as an application to health monitoring on social media over time. In particular, we use perplexity for comparing performance of various topic models in their ability to detect transition and topic predictions in health documents.

Chapter 5

Jointly Learning embeddings and user preference through implicit feedback

5.1 Introduction

In recommender systems, latent factor models have had quite a bit of success. In these models user and items are often represented as latent vectors. In recent years, using embeddings as latent vectors have become popular. Additionally, learning to rank based methods have quite a bit of success in recommender systems. In this work, we are interested in the learning of user preferences, mostly, provided in the form of implicit feedback in RS. Our aim is twofold and concerns:

- the development of a theoretical framework for learning user preference in recommender systems and its analysis in the worst case where all users provide a minimum of positive/negative feedback;
- the design of a new neural-network model based on this framework that learns the preference of users over pairs of items and their representations in an embedded space simultaneously without requiring any contextual information.

In Section 5.2, we first describe the theoretical study of the work. Then, in Section 5.3, we describe a neural network to learn representation and pairwise ranking objective function simultaneously. Then, finally in Section 5.4, we study how the neural network can be extended to handle diversity. This chapter is under review for a publication.

5.2 Theoretical Study

We denote by $\mathcal{U} \subseteq \mathbb{N}$ (resp. $\mathcal{I} \subseteq \mathbb{N}$) the set of indexes over users (resp. the set of indexes over items). Further, for each user $u \in \mathcal{U}$, we consider two subsets of items $\mathcal{I}_u^- \subset \mathcal{I}$ and $\mathcal{I}_u^+ \subset \mathcal{I}$ such that;

$$i) \mathcal{I}_u^- \neq \emptyset \text{ and } \mathcal{I}_u^+ \neq \emptyset,$$

- $$ii) \text{ for any pair of items } (i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-; u \text{ has a preference, symbolized by } \succ_u. \\ \text{Hence } i \succ_u i' \text{ implies that, user } u \text{ prefers item } i \text{ over item } i'.$$

From this preference relation, a desired output $y_{i,u,i'} \in \{-1, +1\}$ is defined over each

triplet $(i, u, i') \in \mathcal{J}_u^+ \times \mathcal{U} \times \mathcal{J}_u^-$ as:

$$y_{i,u,i'} = \begin{cases} 1 & \text{if } i \succ_u i', \\ -1 & \text{otherwise.} \end{cases} \quad (5.2.1)$$

Learning objective

The learning task we address is to find a scoring function f from the class of functions $\mathcal{F} = \{f \mid f : \mathcal{J} \times \mathcal{U} \times \mathcal{J} \rightarrow \mathbb{R}\}$ that minimizes the ranking loss:

$$\mathcal{L}(f) = \mathbb{E} \left[\frac{1}{|\mathcal{J}_u^+| |\mathcal{J}_u^-|} \sum_{i \in \mathcal{J}_u^+} \sum_{i' \in \mathcal{J}_u^-} \mathbb{1}_{y_{i,u,i'} f(i,u,i') < 0} \right], \quad (5.2.2)$$

where $|\cdot|$ measures the cardinality of sets and $\mathbb{1}_\pi$ is the indicator function which is equal to 1, if the predicate π is true, and 0 otherwise. Here we suppose that there exists a mapping function $\Phi : \mathcal{U} \times \mathcal{J} \rightarrow \mathcal{X} \subseteq \mathbb{R}^k$ that projects a pair of user and item indices into a feature space of dimension k , and a function $g : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that each function $f \in \mathcal{F}$ can be decomposed as:

$$\forall u \in \mathcal{U}, (i, i') \in \mathcal{J}_u^+ \times \mathcal{J}_u^-, f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')). \quad (5.2.3)$$

In the next section we will present a Neural-Network model that learns the mapping function Φ and outputs the function f based on a non-linear transformation of the user-item feature representation, defining the function g .

The previous loss (5.2.2) is a pairwise ranking loss and it is related to the Area under the ROC curve [Usunier et al., 2005]. The learning objective is, hence, to find a function f from the class of functions \mathcal{F} with a small expected risk, by minimizing the empirical error over a training set

$$S = \{(\mathbf{z}_{i,u,i'} \doteq (i, u, i'), y_{i,u,i'}) \mid u \in \mathcal{U}, (i, i') \in \mathcal{J}_u^+ \times \mathcal{J}_u^-\},$$

constituted over N users, $\mathcal{U} = \{1, \dots, N\}$, and their respective preferences over M

items, $\mathcal{J} = \{1, \dots, M\}$ and is given by:

$$\begin{aligned}\hat{\mathcal{L}}(f, S) &= \frac{1}{N} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{J}_u^+| |\mathcal{J}_u^-|} \sum_{i \in \mathcal{J}_u^+} \sum_{i' \in \mathcal{J}_u^-} \mathbb{1}_{y_{i,u,i'}(f(i,u,i')) < 0} \\ &= \frac{1}{N} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{J}_u^+| |\mathcal{J}_u^-|} \sum_{i \in \mathcal{J}_u^+} \sum_{i' \in \mathcal{J}_u^-} \mathbb{1}_{y_{i,u,i'}(g(\Phi(u,i)) - g(\Phi(u,i'))) < 0}.\end{aligned}\quad (5.2.4)$$

However this minimization problem involves dependent random variables as for each user u and item i ; all comparisons $g(\Phi(u,i)) - g(\Phi(u,i'))$; $i' \in \mathcal{J}_u^-$ involved in the empirical error (5.2.4) share the same observation $\Phi(u,i)$. Different studies proposed generalization error bounds for learning with interdependent data [Amini and Usunier, 2015]. Among the prominent works that address this problem are a series of contributions based on the idea of graph coloring introduced in [Janson, 2004], and which consists in dividing a graph $\Omega = (\mathcal{V}, \mathcal{E})$ that links dependent variables represented by its nodes \mathcal{V} into J sets of *independent* variables, called the exact proper fractional cover of Ω and defined as:

Definition 1 (Exact proper fractional cover of Ω , [Janson, 2004]). Let $\Omega = (\mathcal{V}, \mathcal{E})$ be a graph. $\mathcal{C} = \{(\mathcal{M}_j, \omega_j)\}_{j \in \{1, \dots, J\}}$, for some positive integer J , with $\mathcal{M}_j \subseteq \mathcal{V}$ and $\omega_j \in [0, 1]$ is an exact proper fractional cover of Ω , if: i) it is *proper*: $\forall j$, \mathcal{M}_j is an *independent set*, i.e., there is no connections between vertices in \mathcal{M}_j ; ii) it is an *exact fractional cover* of Ω : $\forall v \in \mathcal{V}$, $\sum_{j: v \in \mathcal{M}_j} \omega_j = 1$.

The weight $W(\mathcal{C})$ of \mathcal{C} is given by: $W(\mathcal{C}) \doteq \sum_{j=1}^J \omega_j$ and the minimum weight $\chi^*(\Omega) = \min_{\mathcal{C} \in \mathcal{K}(\Omega)} W(\mathcal{C})$ over the set $\mathcal{K}(\Omega)$ of all exact proper fractional covers of Ω is the *fractional chromatic number* of Ω .

Figure 5.1 depicts an exact proper fractional cover corresponding to the problem we consider for a toy problem with $M = 1$ user u , and $|\mathcal{J}_u^+| = 2$ items preferred over $|\mathcal{J}_u^-| = 3$ other ones. In this case, the nodes of the dependency graph correspond to 6 pairs constituted by; pairs of the user and each of the preferred items, with the pairs constituted by the user and each of the no preferred items, involved in the empirical loss (5.2.4). Among all the sets containing independent pairs of examples, the one shown in Figure 5.1,(c) is the exact proper fractional cover of the Ω and the fractional chromatic number is in this case $\chi^*(\Omega) = |\mathcal{J}_u^-| = 3$.

By mixing the idea of graph coloring with the Laplace transform, Hoeffding like

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

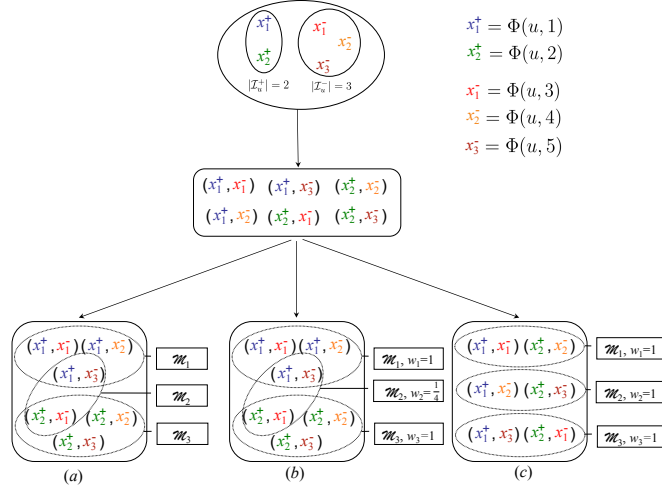


Figure 5.1 – A toy problem with 1 user who prefers $|\mathcal{I}_u^+| = 2$ items over $|\mathcal{I}_u^-| = 3$ other ones (top). The dyadic representation of pairs constituted with the representation of the user and each of the representations of preferred and non-preferred items (middle). Different covering of the dependent set, (a) and (b); as well as the exact proper fractional cover, (c), corresponding to the smallest disjoint sets containing independent pairs.

concentration inequalities for the sum of dependent random variables are proposed by [Janson, 2004]. In [Usunier et al., 2006] this result is extended to provide a generalization of the bounded differences inequality of [McDiarmid, 1989] to the case of interdependent random variables. This extension then paved the way for the definition of the *fractional Rademacher complexity* that generalizes the idea of Rademacher complexity and allows one to derive generalization bounds for scenarios where the training data are made of dependent data.

In the worst case scenario where all users provide the lowest interactions over the items, which constitutes the bottleneck of all recommendation systems:

$$\forall u \in S, |\mathcal{I}_u^-| = n_*^- = \min_{u' \in S} |\mathcal{I}_{u'}^-|, \text{ and } |\mathcal{I}_u^+| = n_*^+ = \min_{u' \in S} |\mathcal{I}_{u'}^+|,$$

the empirical loss (5.2.4) is upper-bounded by:

$$\begin{aligned} \hat{\mathcal{L}}(f, S) &\leq \hat{\mathcal{L}}_*(f, S) \\ &= \frac{1}{N} \frac{1}{n_*^- n_*^+} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u^+} \sum_{i' \in \mathcal{I}_u^-} \mathbb{1}_{y_{i,u,i'} f(i,u,i') < 0}. \end{aligned} \quad (5.2.1)$$

Following [Ralaivola and Amini, 2015, Proposition 4], a generalization error bound can be derived for the second term of the inequality above based on local Rademacher Complexities that implies second-order (i.e. variance) information inducing faster convergence rates.

For sake of presentation and in order to be in line with the learning representations of users and items in an embedded space introduced in Section 5.3, let us consider kernel-based hypotheses with $\kappa : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ a *positive semi-definite* (PSD) kernel and $\Phi : \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{X}$ its associated feature mapping function. Further we consider linear functions in the feature space with bounded norm:

$$\mathcal{G}_B = \{g_w \circ \Phi : (u, i) \in \mathcal{U} \times \mathcal{I} \mapsto \langle w, \Phi(u, i) \rangle \mid \|w\| \leq B\} \quad (5.2.2)$$

where w is the weight vector defining the kernel-based hypotheses and $\langle \cdot, \cdot \rangle$ denotes the dot product. We further define the following associated function class:

$$\mathcal{F}_B = \{z_{i,u,i'} \doteq (i, u, i') \mapsto g_w(\Phi(u, i)) - g_w(\Phi(u, i')) \mid g_w \in \mathcal{G}_B\},$$

and the parameterized family $\mathcal{F}_{B,r}$ which, for $r > 0$, is defined as:

$$\mathcal{F}_{B,r} = \{f : f \in \mathcal{F}_B, \mathbb{V}[f] \doteq \mathbb{V}_{z,y}[\mathbb{1}_{yf(z)}] \leq r\},$$

where $\mathbb{V}[\cdot]$ denotes the variance. The fractional Rademacher complexity introduced in [Usunier et al., 2006] entails our analysis:

$$\mathfrak{R}_S(\mathcal{F}) = \frac{2}{m} \mathbb{E}_\xi \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \sup_{f \in \mathcal{F}} \sum_{\substack{\alpha \in \mathcal{M}_j \\ z_\alpha \in S}} \xi_\alpha f(z_\alpha),$$

where $m = N \times n_*^+ \times n_*^-$ is the total number of triplets z in the training set and $(\xi_i)_{i=1}^m$ is a sequence of independent Rademacher variables verifying $\mathbb{P}(\xi_i = 1) = \mathbb{P}(\xi_i = -1) = \frac{1}{2}$.

Theorem 1. *Let \mathcal{U} be a set of M independent users, such that each user $u \in \mathcal{U}$ prefers n_*^+ items over n_*^- ones in a predefined set of \mathcal{I} items. Let $S = \{(z_{i,u,i'} \doteq (i, u, i'), y_{i,u,i'}) \mid u \in \mathcal{U}, (i, i') \in \mathcal{I}_u^+ \times \mathcal{I}_u^-\}$ be the associated training set, then for any*

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

$1 > \delta > 0$ the following generalization bound holds for all $f \in \mathcal{F}_{B,r}$ with probability at least $1 - \delta$:

$$\begin{aligned} \mathcal{L}(f) &\leq \hat{\mathcal{L}}_*(f, S) + \frac{2B\mathfrak{C}(S)}{Nn_*^+} + \\ &\quad \frac{5}{2} \left(\sqrt{\frac{2B\mathfrak{C}(S)}{Nn_*^+}} + \sqrt{\frac{r}{2}} \right) \sqrt{\frac{\log \frac{1}{\delta}}{n_*^+}} + \frac{25 \log \frac{1}{\delta}}{48 n_*^+}, \end{aligned}$$

where $\mathfrak{C}(S) = \sqrt{\frac{1}{n_*} \sum_{j=1}^{n_*} \mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha) \right]}$, $\mathbf{z}_\alpha = (i_\alpha, u_\alpha, i'_\alpha)$ and

$$\begin{aligned} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha) &= \kappa(\Phi(u_\alpha, i_\alpha), \Phi(u_\alpha, i_\alpha)) \\ &+ \kappa(\Phi(u_\alpha, i'_\alpha), \Phi(u_\alpha, i'_\alpha)) - 2\kappa(\Phi(u_\alpha, i_\alpha), \Phi(u_\alpha, i'_\alpha)). \end{aligned}$$

Proof. As the set of users \mathcal{U} is supposed to be independent, the exact fractional cover of the dependency graph corresponding to the training set S will be the union of the exact fractional cover associated to each user such that cover sets which do not contain any items in common are joined together.

Following [Ralaivola and Amini, 2015, Proposition 4], for any $1 > \delta > 0$ we have with probability at least $1 - \delta$:

$$\begin{aligned} &\mathbb{E}_S[\hat{\mathcal{L}}_*(f, S)] - \hat{\mathcal{L}}_*(f, S) \\ &\leq \inf_{\beta > 0} \left((1 + \beta)\mathfrak{R}_S(\mathcal{F}_{B,r}) + \frac{5}{4} \sqrt{\frac{2r \log \frac{1}{\delta}}{n_*^+}} + \frac{25}{16} \left(\frac{1}{3} + \frac{1}{\beta} \right) \frac{\log \frac{1}{\delta}}{n_*^+} \right) \end{aligned}$$

The infimum is reached for $\beta^* = \sqrt{\frac{25}{16} \frac{\log \frac{1}{\delta}}{n_*^+ \times \mathfrak{R}_S(\mathcal{F}_{B,r})}}$ which by plugging it back into the upper-bound, and from equation (5.2.1), gives:

$$\mathcal{L}(f) \leq \hat{\mathcal{L}}_*(f, S) + \mathfrak{R}_S(\mathcal{F}_{B,r}) + \frac{5}{2} \left(\sqrt{\mathfrak{R}_S(\mathcal{F}_{B,r})} + \sqrt{\frac{r}{2}} \right) \sqrt{\frac{\log \frac{1}{\delta}}{n_*^+}} + \frac{25 \log \frac{1}{\delta}}{48 n_*^+}. \quad (5.2.3)$$

Now, for all $j \in \{1, \dots, J\}$ and $\alpha \in \mathcal{M}_j$, let (u_α, i_α) and (u_α, i'_α) be the first and the second pair constructed from \mathbf{z}_α , then from the bilinearity of dot product and the

Cauchy-Schwartz inequality, $\mathfrak{R}_S(\mathcal{F}_{B,r})$ is upper-bounded by:

$$\begin{aligned}
& \frac{2}{m} \mathbb{E}_\xi \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \sup_{f \in \mathcal{F}_{B,r}} \left\langle \mathbf{w}, \sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} \xi_\alpha (\Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha)) \right\rangle \\
& \leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \mathbb{E}_{\mathcal{M}_j} \mathbb{E}_\xi \left\| \sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} \xi_\alpha (\Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha)) \right\| \\
& \leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j} \mathbb{E}_\xi \left[\sum_{\substack{\alpha, \alpha' \in \mathcal{M}_j \\ \mathbf{z}_\alpha, \mathbf{z}_{\alpha'} \in S}} \xi_\alpha \xi_{\alpha'} d(\mathbf{z}_\alpha, \mathbf{z}_{\alpha'}) \right] \right)^{1/2}, \tag{5.2.4}
\end{aligned}$$

where the last inequality follows from Jensen's inequality and the concavity of the square root, and

$$d(\mathbf{z}_\alpha, \mathbf{z}_{\alpha'}) = \langle \Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha), \Phi(u_\alpha, i_\alpha) - \Phi(u_\alpha, i'_\alpha) \rangle.$$

Further, for all $j \in \{1, \dots, n_*^-\}$, $\alpha, \alpha' \in \mathcal{M}_j$, $\alpha \neq \alpha'$; we have $\mathbb{E}_\xi[\xi_\alpha \xi_{\alpha'}] = 0$, [Shawe-Taylor and Cristianini, 2004, p. 91] so:

$$\begin{aligned}
\mathfrak{R}_S(\mathcal{F}_{B,r}) & \leq \frac{2B}{m} \sum_{j=1}^{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha) \right] \right)^{1/2} \\
& = \frac{2B n_*^-}{m} \sum_{j=1}^{n_*^-} \frac{1}{n_*^-} \left(\mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha) \right] \right)^{1/2}.
\end{aligned}$$

By using Jensen's inequality and the concavity of the square root once again, we finally get

$$\mathfrak{R}_S(\mathcal{F}_{B,r}) \leq \frac{2B}{N n_*^+} \sqrt{\sum_{j=1}^{n_*^-} \frac{1}{n_*^-} \mathbb{E}_{\mathcal{M}_j} \left[\sum_{\substack{\alpha \in \mathcal{M}_j \\ \mathbf{z}_\alpha \in S}} d(\mathbf{z}_\alpha, \mathbf{z}_\alpha) \right]}. \tag{5.2.5}$$

The result follows from equations (5.2.3) and (5.2.5). \square

This result suggests that :

- even though the training set S contains interdependent observations; following [Vapnik, 2000, theorem 2.1, p. 38], theorem 1 gives insights on the consistency of the empirical risk minimization principle with respect to (5.2.1),
- in the case where the feature space $\mathcal{X} \subseteq \mathbb{R}^k$ is of finite dimension; lower values of k involves lower kernel estimation and hence lower complexity term $\mathfrak{C}(S)$ which implies a tighter generalization bound.

5.3 A Neural Network model to learn user preference

Some studies proposed to find the dyadic representation of users and items in an embedded space, using neighborhood similarity information [Volkovs and Yu, 2015] or the Bayesian Personalized Ranking (BPR) [Rendle et al., 2009]. In this section, we propose a feed-forward Neural Network, denoted as NERVE, to jointly learn the embedding representation, $\Phi(\cdot)$, as well as the scoring function, $f(\cdot)$, defined previously. The input of the network is a triplet (i, u, i') composed by the indexes of an item i , a user u and a second item i' ; such that the user u has a preference over the pair of items (i, i') expressed by the desired output $y_{i,u,i'}$, defined with respect to the preference relation \succ_u (Eq. 5.2.1). Each index in the triplet is then transformed to a corresponding binary indicator vector \mathbf{i} , \mathbf{u} , and \mathbf{i}' having all its characteristics equal to 0 except the one that indicates the position of the user or the items in its respective set, which is equal to 1. Hence, the following one-hot vector corresponds to the binary vector representation of user $u \in \mathcal{U}$ as shown in Figure 5.2:

$$\mathbf{u}^\top = (0, \dots, 0, \overset{1}{\downarrow}, \overset{u-1}{\downarrow}, \overset{u}{\downarrow}, \overset{u+1}{\downarrow}, \dots, \overset{N}{\downarrow}).$$

Figure 5.2 – one-hot vector corresponds to the binary vector representation of user $u \in \mathcal{U}$.

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

The network then entails three successive layers, namely *Embedding* (SG), *Mapping* and *Dense* hidden layers depicted in Figure 5.3.

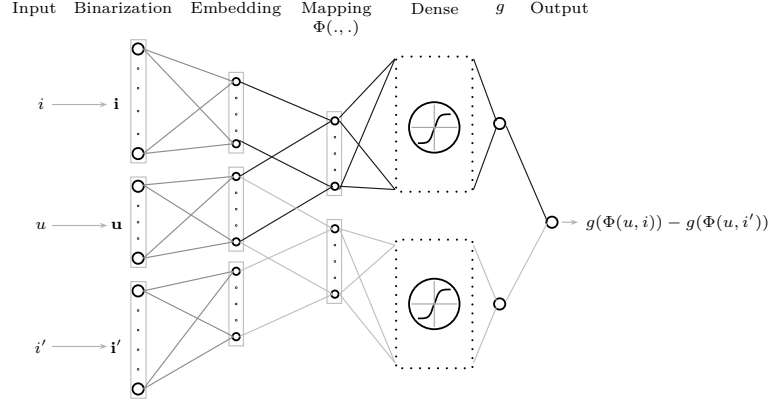


Figure 5.3 – The architecture of NERVvE trained to reflect the preference of a user u over a pair of items i and i' .

- The *Embedding* layer transforms the sparse binary representations of the user and each of the items to a denser real-valued vectors. We denote by \mathbf{U}_u and \mathbf{V}_i the transformed vectors of user u and item i ; and $\mathbf{U} = (\mathbf{U}_u)_{u \in \mathcal{U}}$ and $\mathbf{V} = (\mathbf{V}_i)_{i \in \mathcal{I}}$ the corresponding matrices. Note that as the binary indicator vectors of users and items contain one single non-null characteristic, each entry of the corresponding dense vector in the SG layer is connected by only one weight to that characteristic.
- The *Mapping* layer is composed of two groups of units each being obtained from the element-wise product between the user representation vector \mathbf{U}_u of a user u and a corresponding item representation vector \mathbf{V}_i of an item i inducing the feature representation of the pair (u, i) ; $\Phi(u, i)$.
- Each of these units are also fully connected to the units of a *Dense* layer composed of successive hidden layers (see Section 6.3 for more details related to the number of hidden units and the activation function used in this layer).

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

The model is trained such that the output of each of the dense layers reflects the relationship between the corresponding item and the user and is mathematically defined by a multivariate real-valued function $g(\cdot)$. Hence, for an input (i, u, i') , the output of each of the dense layers is a real-value score that reflects a preference associated to the corresponding pair (u, i) or (u, i') (i.e. $g(\Phi(u, i))$ or $g(\Phi(u, i'))$). Finally, the prediction given by NERVE for an input (i, u, i') is:

$$f(i, u, i') = g(\Phi(u, i)) - g(\Phi(u, i')). \quad (5.3.1)$$

Algorithmic implementation

We decompose the ranking loss as a linear combination of two logistic surrogates:

$$\mathcal{L}_{c,p}(f, \mathbb{U}, \mathbb{V}, \mathcal{S}) = \mathcal{L}_c(f, \mathcal{S}) + \mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}), \quad (5.3.2)$$

where the first term reflects the ability of the non-linear transformation of user and item feature representations, $g(\Phi(\cdot, \cdot))$, to respect the relative ordering of items with respect to users' preferences:

$$\mathcal{L}_c(f, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \log(1 + e^{y_{i,u,i'}(g(\Phi(u,i')) - g(\Phi(u,i')))}). \quad (5.3.3)$$

The second term focuses on the quality of the compact dense vector representations of items and users that have to be found, as measured by the ability of the dot-product in the resulting embedded vector space to respect the relative ordering of preferred items by users:

$$\mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{z}_{i,u,i'}, y_{i,u,i'}) \in \mathcal{S}} \left[\log(1 + e^{y_{i,u,i'} \mathbf{U}_u^\top (\mathbf{V}_{i'} - \mathbf{V}_i)}) + \lambda (\|\mathbf{U}_u\| + \|\mathbf{V}_{i'}\| + \|\mathbf{V}_i\|) \right], \quad (5.3.4)$$

where λ is a regularization parameter for the user and items norms. Finally, one can also consider a version in which both losses are assigned different weights:

$$\mathcal{L}_{c,p}(f, \mathbb{U}, \mathbb{V}, \mathcal{S}) = \alpha \mathcal{L}_c(f, \mathcal{S}) + (1 - \alpha) \mathcal{L}_p(\mathbb{U}, \mathbb{V}, \mathcal{S}), \quad (5.3.5)$$

where $\alpha \in [0, 1]$ is a real-valued parameter to balance between ranking prediction

ability and expressiveness of the learned item and user representations. Both options will be discussed in the experimental section.

Training phase

The training of the NERVE is done by back-propagating [Bottou, 2012] the error-gradients from the output to both the deep and embedding parts of the model using mini-batch stochastic optimization (Algorithm 1).

During training, the input layer takes a random set $\tilde{\mathcal{S}}_n$ of size n of interactions by building triplets (i, u, i') based on this set, and generating a sparse representation from id's vector corresponding to the picked user and the pair of items. The binary vectors of the examples in $\tilde{\mathcal{S}}_n$ are then propagated throughout the network, and the ranking error (Eq. 5.3.2) is back-propagated.

Algorithm 2 NERVE: Learning phase

Require:

T : maximal number of epochs

A set of users $\mathcal{U} = \{1, \dots, N\}$

A set of items $\mathcal{I} = \{1, \dots, M\}$

for $ep = 1, \dots, T$ **do**

 Randomly sample a mini-batch $\tilde{\mathcal{S}}_n \subseteq \mathcal{S}$ of size n from the original user-item matrix

for all $((i, u, i'), y_{i,u,i'}) \in \tilde{\mathcal{S}}_n$ **do**

Propagate (i, u, i') from the input to the output.

Retro-propagate the pairwise ranking error (Eq. 5.3.2) estimated over $\tilde{\mathcal{S}}_n$.

Ensure: Users and items latent feature matrices \mathbb{U}, \mathbb{V} and the model weights.

Model Testing

As for the prediction phase, shown in Algorithm 2, a ranked list $\mathfrak{N}_{u,k}$ of the $k \ll M$ preferred items for each user in the test set is maintained while retrieving the set \mathcal{I} . Given the latent representations of the triplets, and the weights learned; the two first items in \mathcal{I} are placed in $\mathfrak{N}_{u,k}$ in a way which ensures that preferred one, i^* , is in the first position. Then, the algorithm retrieves the next item, $i \in \mathcal{I}$ by comparing it to i^* . This step is simply carried out by comparing the model's output over the concatenated binary indicator vectors of (i^*, u, i) and (i, u, i^*) .

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

Hence, if $f(i, u, i^*) > f(i^*, u, i)$, which from Equation 5.3.1 is equivalent to $g(\Phi(u, i)) > g(\Phi(u, i^*))$, then i is predicted to be preferred over i^* ; $i \succ_u i^*$; and it is put at the first place instead of i^* in $\mathfrak{N}_{u,k}$. Here we assume that the predicted preference relation \succ_u is transitive, which then ensures that the predicted order in the list is respected. Otherwise, if i^* is predicted to be preferred over i , then i is compared to the second preferred item in the list, using the model's prediction as before, and so on. The new item, i , is inserted in $\mathfrak{N}_{u,k}$ in the case if it is found to be preferred over another item in $\mathfrak{N}_{u,k}$.

By repeating the process until the end of \mathcal{I} , we obtain a ranked list of the k most preferred items for the user u . Algorithm 2 does not require an ordering of the whole set of items, as also in most cases we are just interested in the relevancy of the top ranked items for assessing the quality of a model. Further, its complexity is at most $O(k \times M)$ which is convenient in the case where $M \gg 1$. The merits of a similar algorithm have been discussed by [Ailon and Mohri, (2008)] but, as pointed out above, the basic assumption for inserting a new item in the ranked list $\mathfrak{N}_{u,k}$ is that the predicted preference relation induced by the model should be transitive, which may not hold in

Algorithm 3 NERVE: Testing phase

Require:

A user $u \in \mathcal{U}$; A set of items $\mathcal{I} = \{1, \dots, M\}$;

A set containing the k preferred items in \mathcal{I} by u ;

$\mathfrak{N}_{u,k} \leftarrow \emptyset$;

The output of NERVE, learned over a training set: f

Apply f to the first two items of \mathcal{I} and, note the preferred one i^* and place it at the top of $\mathfrak{N}_{u,k}$;

for $i = 3, \dots, M$ **do**

if $g(\Phi(u, i)) > g(\Phi(u, i^*))$ **then**

 Add i to $\mathfrak{N}_{u,k}$ at rank 1

else

$j \leftarrow 1$

while $j \leq k$ AND $g(\Phi(u, i)) < g(\Phi(u, i_g))$ // where $i_g = \mathfrak{N}_{u,k}(j)$ **do**

$j \leftarrow j + 1$

if $j \leq k$ **then**

 Insert i in $\mathfrak{N}_{u,k}$ at rank j

Ensure: $\mathfrak{N}_{u,k}$;

general.

In our experiments, we also tested a more conventional inference algorithm, which for a given user u , consists in the ordering of items in \mathcal{I} with respect to the output given by the function g , and we did not find any substantial difference in the performance of NERvE_c .

5.4 Diversity

Additionally, we demonstrate how PANDOR can be of a great interest for developing novel algorithms incorporating diversity in RS, where the feedback provided are implicit and no meta information about the proposed items is available. Indeed, although, the goal of a RS is to have fewer flops on the top of the recommended list, inducing more diversity in this recommended list ensures that user may prefer to interact with at least some items in contrast to the situation where we introduce just monotonous relevant items. In addition, the recent work of [Abdollahpouri et al., 2017] shows that diversity can be used in order to control the popularity bias in such type of data, also known as the problem of long tail i.e. a situation where a large majority of items have only very few ratings or clicks, either because they are new or just unpopular.

5.4.1 Incorporating diversity to handle popularity bias in recommender systems

Hereafter, we propose to explore the ability of diversity in RS to overcome the strong bias induced by popular items, or items with high CTR. Also, we focus only on the setting in which we test on all items as most approaches fail to provide good results on such setting. To this end, we propose to evaluate two approaches. The first one was initially proposed by [Wasilewski and Hurley, 2016a] and consider the objective function of Rank-ALS [Takács and Tikk, 2012] augmented with a regularization term that consists of the intra-list diversity (ILD) measure. Then, without loss of generality, we propose to build upon NERvE_c . The diversity regularizers, we add here for RankALS or NERvE_c , can be used with any loss function. In [Wasilewski and Hurley, 2016a], the authors used the movies' genre to compute distances between two items. However,

5. JOINTLY LEARNING EMBEDDINGS AND USER PREFERENCE THROUGH IMPLICIT FEEDBACK

on many occasions item metadata is not available. To overcome this problem of absence of item metadata, we propose to compute item embeddings as meta data [Barkan and Koenigstein, 2016a]. Here, we would like to stress on the fact that computing embeddings with the Item2Vec [Barkan and Koenigstein, 2016a] technique to measure diversity is a fresh departure from previous works on this topic; indeed, in our case, item diversity is not related to the characteristics of the items themselves, such as the genre, or the category, but rather to the diversity of the sequence of items displayed to users. This means that our goal is to, somehow, force the RS algorithm to display various diverse sequences of items to each user. We compute item embeddings, with Gensims based Skip-Gram implementation of Word2Vec (adapted to Item2Vec). We set the dimension to 20 and consider 3 as the context window.

NERvE_c with diversity For NERvE_c, we propose to minimize the objective function of NERvE_c and to incorporate diversity within the list of items recommended to each user through a penalty term based on the Kullback-Leibler (KL). To this end, we propose to measure the dissimilarities between each pair of items $i \in \mathcal{S}_u$ (where \mathcal{S}_u^k denotes the list of items and k its size) of the loss function associated to this new problem can be written as

$$\mathcal{L}_{\text{NERvE}_c}(f, U, V, \mathcal{S}) + \beta \frac{1}{|U|} \sum_{u \in U} \left(\frac{1}{k(k-1)} \right) \sum_{i, i' \in \mathcal{S}_u^k} KL(\mathbf{V}_i^{\ell_1} || \mathbf{V}_{i'}^{\ell_1}),$$

where $\mathbf{V}_i^{\ell_1}$ (resp. $\mathbf{V}_{i'}^{\ell_1}$) is the ℓ_1 -normalized embedding associated with item i (resp. i'); β is the diversity inducing regularization parameter whose role is to induce more or less diversity in the final list of recommended items. Positive values of β imply minimizing diversity and vice versa. We cross-validate the value of β on a validation set built from the original training set.

5.5 Conclusion

We presented and analyzed a learning to rank framework for recommender systems which consists of learning user preferences over items. We showed that the minimization of pairwise ranking loss over user preferences involves dependent random variables and provided a theoretical analysis by proving the consistency of the empirical

risk minimization in the worst case where all users choose a minimal number of positive and negative items. From this analysis, we then proposed NERvE , a new neural-network based model for learning the user preference, where both the user's and item's representations and the function modeling the user's preference over pairs of items are learned simultaneously. The learning phase is guided using a ranking objective that can capture the ranking ability of the prediction function as well as the expressiveness of the learned embedded space, where the preference of users over items is respected by the dot product function defined over that space. The training of NERvE is carried out using the back-propagation algorithm in mini-batches defined over a user-item matrix containing implicit information in the form of subsets of preferred and non-preferred items. The learning capability of the model over both prediction and representation problems show their interconnection and also that the proposed double ranking objective allows to conjugate them well. Finally, we proposed an objective function which extends objective function of NERvE_c which incorporates diversity within the list of items recommended to each user through a penalty term based on the Kullback-Leibler (KL).

Chapter 6

Experimental Results

6.1 Introduction

This Chapter details the results of all the experiments which we conducted in order to depict the efficacy of the models we developed and datasets we contributed. By efficacy of datasets, we mean that the datasets are well suited for setting benchmarks for recommender models developed for leveraging implicit feedback.

In Section 6.3, we first show the performance of NERvE on many implicit feedback datasets as compared to other state-of-the-art methods in RS for implicit feedback. Then, in Section 6.4, we present the first results of state-of-the-art models using contextual information on KASANDR, the data-set which we contributed. We proceed by presenting the results that we obtained on PANDOR in Section 6.4. In particular, we demonstrate how the performance of the baselines gets strongly affected due to popularity bias in the dataset, and how by introducing diversity in Section 6.4 we can overcome this problem. Later in the same Section 6.4, we show how introducing contextual information such as topics (extracted from TM-LDA) can boost the performance of Factorization Machines (a popular baseline for RS).

We run all experiments on a cluster of five 32 core Intel Xeon @ 2.6Ghz CPU (with 20MB cache per core) systems with 256 Giga RAM running Debian GNU/Linux 8.6 (wheezy) operating system.

6.2 Baselines and Evaluation Protocol

We describe all the approaches we are going to test against. We choose three non-machine learning approaches:

- The random rule (Rand), that consists of recommending random items to the user,
- The popularity rule (Pop), that consists of recommending items with the best degree of success among all users,
- the past interaction technique (PastI), that consists of recommending items that the user has already interacted with.

6. EXPERIMENTAL RESULTS

We implement our version of Pop, PastI and Rand. We choose various machine learning approaches which are meant for handling implicit feedback in RS as follows:

- Matrix Factorization (MF) [Koren et al., 2009]. For MF, we use built-in implementation of Spark which is based on [Hu et al., 2008a].
- Factorization Machines (FM) [Rendle, 2010]. In terms of implementation, we use LIBFM
- Field-Aware Factorization Machines (FFM) [Juan et al., 2016a]. FFM has won two recent world-wide click-through rate prediction competitions (hosted by Criteo and Avazu). In terms of implementation, we use LIBFFM for FFM, respectively.
- Rank-ALS [Takács and Tikk, 2012], a ranking formulation of Matrix Factorization presented in Section 2.4.2;
- Bayesian Personalized Ranking (BPR) [Rendle et al., 2009], a pairwise ranking approach; provides an optimization criterion based on implicit feedback; which is the maximum posterior estimator derived from a Bayesian analysis of the pairwise ranking problem, and proposes an algorithm based on Stochastic Gradient Descent to optimize it. The model can further be extended to the explicit feedback case.
- LightFM [Kula, 2015], that relies on learning the embedding of users and items with the Skip-gram model while optimizing a ranking loss. LightFM was first proposed to deal with the problem of cold-start using meta information. As with our approach, it relies on learning the embedding of users and items with the Skip-gram model and optimizes the cross-entropy loss.
- Co-Factor [Liang et al., 2016], is a model for implicit feedback, constraints the objective of matrix factorization to jointly use item representations with a factorized shifted positive pointwise mutual information matrix of item co-occurrence counts. The model was found to outperform WMF [Hu et al., 2008b] also proposed for implicit feedback.

- $\text{NERvE}_{c,p}$ uses a linear combination of \mathcal{L}_p and \mathcal{L}_c as the objective function, with $\alpha \in]0, 1[$. We study the two situations presented before (w.r.t. the presence/absence of a supplementary weighting hyper-parameter).
- NERvE_p focuses on the quality of the latent representation of users and items by learning the preference and the representation through the ranking loss \mathcal{L}_p (Eq. 5.3.4).
- NERvE_c focuses on the accuracy of the score obtained at the output of the framework and therefore learns the preference and the representation through the ranking loss \mathcal{L}_c (Eq. 5.3.3).

Please note that we show results on subset of baselines in each section wherever it is feasible to run them.

Finally, we consider two settings w.r.t. to the set of items selected for the prediction.

1. Item recommendation only relies on past interacted offers, that is, we only consider for a given user, the items that the user interacted with in the training phase. By interacted, we mean the user was either shown the offer or user clicked on the offer. In the context of movie recommendation, a shown item is defined as a movie for which the given user provided a rating. For KASANDR, PANDOR and KASANDR-GER, the definition is quite straight-forward as the data were collected from an on-line advertising platform, where the items are displayed to the users, who can either click or ignore them. While this is probably the most popular setting in the literature, it is also the less realistic one, as in an real online setting one has to consider all the available items when making prediction.
2. The RS considers the full set of items as possible candidate for the prediction.

The first setting is arguably the most common in academic research, but is abstracted from the real-world problem as at the time of making the recommendation, the notion of shown items is not available, therefore forcing the RS to consider the set of all items as potential candidates. The goal of the second setting is to reflect this real-world scenario, and we can expect lower results as compared to the first setting as the size of the search space of items increases considerably. To summarize, predicting only among the items that were shown to user evaluates the model’s capability of retrieving

highly rated items among the shown ones, while predicting among all items measures the performance of the model on the basis of its ability to recommend offers which user would like to engage in. We proceed with the presentation of the results obtained with our model, `NERvE` on some popular benchmark datasets.

6.3 `NERvE` Results

We conduct a number of experiments aimed at evaluating how the simultaneous learning of user and item representations, as well as the preferences of users over items can be efficiently handled with `NERvE`. To this end, we consider four real-world benchmarks commonly used for collaborative filtering. We validate our approach with respect to different hyper-parameters that impact the accuracy of the model and compare it with competitive state-of-the-art approaches. All subsequently discussed components were implemented in Python3 using the TensorFlow library with version 1.4.0.¹

Datasets

We report results obtained on three publicly available movie datasets, for the task of personalized top-N recommendation: `MOVIELENS`² 100K (ML-100K), `MOVIELENS` 1M (ML-1M) [Harper and Konstan, 2015], `NETFLIX`³, and one clicks dataset, `KASANDR-GER`⁴ [Sidana et al., 2017], a recently released data set for on-line advertising.

- ML-100K, ML-1M and `NETFLIX` consists of user-movie ratings, on a scale of one to five, collected from a movie recommendation service and the Netflix company. The latter was released to support the Netflix Prize competition⁵. For all three datasets, we only keep users who have rated at least five movies and remove users who gave the same rating for all movies. In addition, for `NETFLIX`, we take a subset of the original data and randomly sample 20% of the users and 20% of the items. In the following experiments, as we only compare with approaches

¹<https://www.tensorflow.org/>.

²<https://movielens.org/>

³<http://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a>

⁴<https://archive.ics.uci.edu/ml/datasets/KASANDR>

⁵B. James and L. Stan, The Netflix Prize (2007).

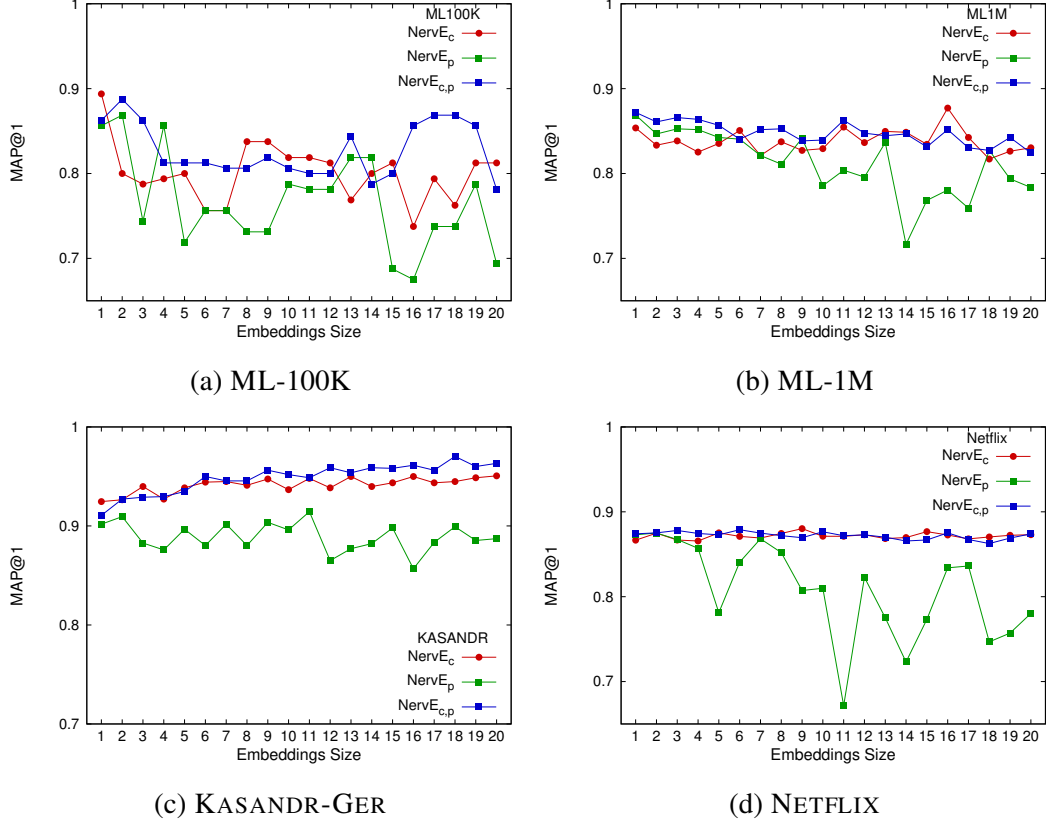


Figure 6.1 – MAP@1 as a function of the dimension of the embedding for ML-100K, ML-1M and KASANDR-GER.

developed for the ranking purposes and our model is designed to handle implicit feedback, these three data sets are made binary such that a rating higher or equal to 4 is set to 1 and to 0 otherwise.

- The original KASANDR dataset contains the interactions and clicks done by the users of Kelkoo, an online advertising platform, across twenty Europeans countries. In this article, we used a subset of KASANDR that only considers interactions from Germany. It gathers 17,764,280 interactions from 521,685 users on 2,299,713 offers belonging to 272 categories and spanning across 801 merchants. For KASANDR-GER, we remove users who gave the same rating for all offers. This implies that all the users who never clicked or always clicked on each and every offer shown to them were removed.

6. EXPERIMENTAL RESULTS

Table 6.1 provides the basic statistics on these collections after pre-processing, as discussed above.

Table 6.1 – Statistics of various collections used in our experiments after preprocessing.

	# of users	# of items	# of interactions	Sparsity
ML-100K	943	1,682	100,000	93.685%
ML-1M	6,040	3,706	1,000,209	95.530%
NETFLIX	90,137	3,560	4,188,098	98.700%
KASANDR-GER	25,848	1,513,038	9,489,273	99.976%

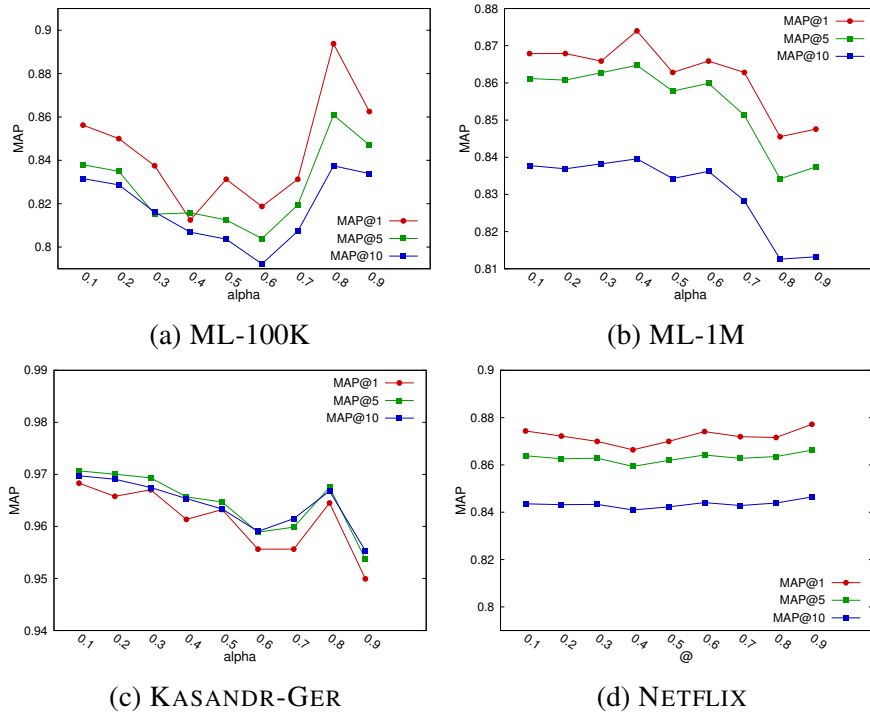


Figure 6.2 – MAP@1, MAP@5, MAP@10 as a function of the value of α for ML-1M, ML-100K and KASANDR-GER.

Compared baselines

In order to validate the framework defined in the previous section, we propose to compare the following approaches: BPR-MF, Co-Factor, LightFM, NERvE_p, NERvE_c, NERvE_{c,p}.

Table 6.2 – Best parameters for NERvE_p , NERvE_c and $\text{NERvE}_{c,p}$ when prediction is done on only shown offers; k denotes the dimension of embeddings, λ the regularization parameter. We also report the number of hidden units per layer.

	ML-100K			ML-1M			NETFLIX			KASANDR-GER		
	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$
k	1	2	2	16	1	1	9	2	6	19	1	18
λ	0.05	0.005	0.005	0.05	0.0001	0.001	0.05	0.01	0.05	0.0001	0.05	0.005
# units	32	64	16	32	16	32	64	16	16	64	16	64

Table 6.3 – Best parameters for NERvE_p , NERvE_c and $\text{NERvE}_{c,p}$ when prediction is done on all offers; k denotes the dimension of embeddings, λ the regularization parameter. We also report the number of hidden units per layer.

	ML-100K			ML-1M			NETFLIX			KASANDR-GER		
	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$	NERvE_c	NERvE_p	$\text{NERvE}_{c,p}$
k	15	5	8	2	11	2	3	13	1	4	16	14
λ	0.001	0.001	0.001	0.05	0.0001	0.001	0.0001	0.001	0.001	0.001	0.0001	0.05
# units	32	16	16	32	64	32	32	64	64	32	64	64

Evaluation protocol

For each dataset, we sort the interactions according to time, and take 80% for training the model and the remaining 20% for testing it. In addition, we remove all users and offers which do not occur during the training phase. We study two different scenarios for the prediction phase: (1) *interacted offers* (2) *all items* as defined in section 6.2.

As a result, in this setting, for ML-100K, ML-1M, KASANDR-GER and NETFLIX, we only consider in average 25, 72, 6 and 8 items for prediction per user.

All comparisons are done based on the Mean Average Precision (MAP) 2.7.2. In the following results, we report MAP at different rank $\ell = 1$ and 10.

Hyper-parameters tuning

First, we provide a detailed study of the impact of the different hyper-parameters involved in the proposed framework NERvE_\cdot . For all datasets, hyper-parameters tuning is done on a separate validation set.

- The size of the embedding is chosen among $k \in \{1, \dots, 20\}$. The impact of k on the performance is presented in Figure 6.1.
- We use ℓ_2 regularization on the embeddings and choose $\lambda \in \{0.0001, 0.001, 0.005, 0.01, 0.05\}$.

6. EXPERIMENTAL RESULTS

- We run NERvE with 1 hidden layer with relu activation functions, where the number of hidden units is chosen in $\{16, 32, 64\}$.
- In order to train NERvE, we use ADAM [Kingma and Ba, 2014] and found the learning rate $\eta = 1e - 3$ to be more efficient for all our settings. For other parameters involved in Adam, i.e., the exponential decay rates for the moment estimates, we keep the default values ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$).
- Finally, we fix the number of epochs to be $T = 10,000$ in advance and the size of mini-batches to $n = 512$.
- One can see that all three versions of NERvE perform the best with a quite small number of hidden units, only one hidden layer and a low dimension for the representation. As a consequence, they involve a few number of parameters to tune while training.
- In terms of the ability to recover a relevant ranked list of items for each user, we also tune the hyper-parameter α (Eq. 5.3.5) which balances the weight given to the two terms in $\text{NERvE}_{c,p}$. These results are shown in Figure 6.2, where the values of α are taken in the interval $[0, 1]$. While it seems to play a significant role on ML-100K and KASANDR-GER, we can see that for ML-1M the results in terms of MAP are stable, regardless the value of α .

From Figure 6.1, when prediction is done on the interacted offers, it is clear that best MAP@1 results are generally obtained with small sizes of item and user embedded vector spaces k . These empirical results support our theoretical analysis where we found that small k induces smaller generalization bounds. This observation on the dimension of embedding is also in agreement with the conclusion of [Kula, 2015], which uses the same technique for representation learning. For instance, one can see that on ML-1M, the highest MAP is achieved with a dimension of embedding equals to 1. Since in the interacted offers setting, the prediction is done among the very few shown offers, NERvE makes non-personalized recommendations. This is due to the fact that having $k = 1$ means that the recommendations for a given user with a positive (negative) value is done by sorting the positive (negative) items according to their learned embeddings, and in some sense, can therefore be seen as a bi-polar

popularity model. This means that in such cases popularity and non-personalized based approaches are perhaps the best way to make recommendations. For reproducibility purpose, we report the best combination of parameters for each variant of NERvE in Table 6.2 and Table 6.3. Hereafter, we compare and summarize the performance of NERvE , with the baseline methods on various data sets. Empirically, we observed that the version of $\text{NERvE}_{c,p}$ where both \mathcal{L}_c and \mathcal{L}_p have an equal weight while training gives better results on average, and we decided to only report these results later.

Tables 6.4 and 6.5 report all results. In addition, in each case, we statistically compare the performance of each algorithm, and we use bold face to indicate the highest performance, and the symbol \downarrow indicates that performance is significantly worst than the best result, according to a Wilcoxon rank sum test used at a p-value threshold of 0.01 [Lehmann and D’Abrera, 2006].

Setting 1 : interacted items

When the prediction is done over offers which user interacted with (Table 6.4), the NERvE architecture, regardless the weight given to α , beats all the other algorithms on KASANDR-GER, ML-100K and ML-1M. However, on NETFLIX, BPR-MF outperforms our approach in terms of MAP@1. This may be owing to the fact that the binarized NETFLIX movie data set is strongly biased towards the popular movies and usually, the majority of users have watched one or the other popular movies in such data sets and rated them well. In NETFLIX, around 75% of the users have given ratings greater to 4 to the top-10 movies. We believe that this phenomenon adversely affects the performance of NERvE . However, on KASANDR-GER, which is the only true implicit dataset NERvE significantly outperforms all other approaches.

Setting 2 : all items

When the prediction is done over all offers (Table 6.5), we can make two observations. First, all the algorithms encounters an extreme drop of their performance in terms of MAP. Second, NERvE framework significantly outperforms all other algorithms on all datasets, and this difference is all the more important on KASANDR-GER, where for instance $\text{NERvE}_{c,p}$ is in average 15 times more efficient. We believe, that our model is a fresh departure from the models which learn pairwise ranking function without the

6. EXPERIMENTAL RESULTS

knowledge of embeddings or which learn embeddings without learning any pairwise ranking function. While learning pairwise ranking function, our model is aware of the learned embeddings so far and vice-versa. We demonstrate that the simultaneous learning of two ranking functions helps in learning hidden features of implicit data and improves the performance of NERvE.

Comparison between NERvE versions

One can note that while optimizing ranking losses by Eq. 5.3.2 or Eq. 5.3.3 or Eq. 5.3.4, we simultaneously learn representation and preference function; the main difference is the amount of emphasis we put in learning one or another. The results presented in both tables tend to demonstrate that, in almost all cases, optimizing the linear combination of the pairwise-ranking loss and the embedding loss (NERvE_{c,p}) indeed increases the quality of overall recommendations than optimizing standalone losses to learn embeddings and pairwise preference function. For instance, when the prediction is done over offers which user interacted with (Table 6.4), (NERvE_{c,p}) outperforms (NERvE_p) and (NERvE_c) on ML-1M, KASANDR-GER and NETFLIX. When prediction is done on all offers (Table 6.5), (NERvE_{c,p}) outperforms (NERvE_p) and (NERvE_c) on KASANDR-GER. Thus, in case of interacted offers setting, optimizing ranking and embedding loss simultaneously boosts performance on all datasets. However, in the setting of all offers, optimizing both losses simultaneously is beneficial in case of true implicit feedback datasets such as KASANDR-GER(recall that all other datasets were synthetically made implicit).

Table 6.4 – Results of all state-of-the-art approaches for implicit feedback when prediction is done only on offers shown to users. The best result is in bold, and a [↓] indicates a result that is statistically significantly worse than the best, according to a Wilcoxon rank sum test with $p < .01$.

	ML-100K		ML-1M		NETFLIX		KASANDR-GER	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
BPR-MF	0.613 [↓]	0.608 [↓]	0.788 [↓]	0.748 [↓]	0.909	0.842 [↓]	0.857 [↓]	0.857 [↓]
LightFM	0.772 [↓]	0.770 [↓]	0.832 [↓]	0.795 [↓]	0.800 [↓]	0.793 [↓]	0.937 [↓]	0.936 [↓]
CoFactor	0.718 [↓]	0.716 [↓]	0.783 [↓]	0.741 [↓]	0.693 [↓]	0.705 [↓]	0.925 [↓]	0.918 [↓]
NERvE _c	0.894	0.848	0.877 [↓]	0.835	0.880 [↓]	0.847	0.958 [↓]	0.963 [↓]
NERvE _p	0.881 [↓]	0.846	0.876 [↓]	0.839	0.875 [↓]	0.844	0.915 [↓]	0.923 [↓]
NERvE _{c,p}	0.888 [↓]	0.842	0.884	0.839	0.879 [↓]	0.847	0.970	0.973

Table 6.5 – Results of all state-of-the-art approaches for recommendation on all implicit feedback data sets when prediction is done on all offers. The best result is in bold, and a \downarrow indicates a result that is statistically significantly worse than the best, according to a Wilcoxon rank sum test with $p < .01$

	ML-100K		ML-1M		NETFLIX		KASANDR-GER	
	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10	MAP@1	MAP@10
BPR-MF	0.140 \downarrow	0.261	0.048 \downarrow	0.097 \downarrow	0.035 \downarrow	0.072 \downarrow	0.016 \downarrow	0.024 \downarrow
LightFM	0.144 \downarrow	0.173 \downarrow	0.028 \downarrow	0.096 \downarrow	0.006 \downarrow	0.032 \downarrow	0.002 \downarrow	0.003 \downarrow
CoFactor	0.056 \downarrow	0.031 \downarrow	0.089 \downarrow	0.033 \downarrow	0.049 \downarrow	0.030 \downarrow	0.002 \downarrow	0.001 \downarrow
NERvE _c	0.106 \downarrow	0.137 \downarrow	0.067 \downarrow	0.093 \downarrow	0.032 \downarrow	0.048 \downarrow	0.049 \downarrow	0.059 \downarrow
NERvE _p	0.239	0.249	0.209	0.220	0.080	0.089	0.100 \downarrow	0.100 \downarrow
NERvE _{c,p}	0.111 \downarrow	0.134 \downarrow	0.098 \downarrow	0.119 \downarrow	0.066 \downarrow	0.087	0.269	0.284

6.4 Results on KASANDR and PANDOR

In this Section, we provide results obtained from baseline methods including non-machine learning approaches and three algorithms that have proven efficient for the recommendation task based on implicit feedback for the two datasets described in Chapter 3.

To proceed, we recall the main characteristics of KASANDR and PANDOR in Table 3.3 and 3.7 respectively.

Results on KASANDR

Compared methods In order to depict the effectiveness of KASANDR as a novel dataset which is suitable to test recommendation models using meta-information and encourage future research on recommendation systems using implicit feedback, we show results of following baselines on KASANDR: Pop, PastI, Rand, MF, FM, FFM, FFM-F. We also perform parameter tuning for the aforementioned machine learning algorithms on a different validation set and report the optimum ones in Table 6.6.

Table 6.6 – Parameters used for compared approaches.

Algorithm	Optimization	#Iterations	#Latent Factors	Learning Rate	Reg Param
MF	ALS	20	50	N.A.	0.01
FM	SGD	10	1,1,10	0.001	0.01
FFM	SGD	15	8	0.2	0.001

Furthermore, because we run the tested approaches per country, we define macro

6. EXPERIMENTAL RESULTS

MAP as:

$$\text{Macro MAP@k} = \frac{1}{|C|} \sum_{c \in C} \text{MAP@k}(c)$$

and micro MAP as:

$$\text{Micro MAP@k} = \sum_{c=1}^C \frac{n_c}{N} \text{MAP@k}(c),$$

where c , n_c and N are the country, number of users in that country and total number of users, respectively. One can observe that Micro MAP takes into account the size of the traffic within each country and gives more weight to bigger countries while Macro MAP simply averages the results obtained for all countries.

Table 6.7 – Comparison between all tested methods in terms of Micro and Macro MAP for non-machine learning based methods. The best results are in bold.

	Rand		Pop		PastI	
	Micro	Macro	Micro	Macro	Micro	Macro
MAP@5	2.41E-6	1.54E-005	0.004	0.004	0.017	0.011
MAP@30	4.25E-6	2.33E-005	0.004	0.005	0.017	0.011
MAP@100	5.64E-6	2.996E-005	0.005	0.005	0.016	0.011

Table 6.8 – Comparison between all tested methods in terms of Micro and Macro MAP for machine learning based methods. The best results are in bold.

	MF		FM		FFM		FFM-F	
	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro
MAP@5	0.044	0.037	0.721	0.814	0.732	0.829	0.760	0.861
MAP@30	0.044	0.037	0.726	0.817	0.736	0.831	0.764	0.862
MAP@100	0.044	0.037	0.726	0.817	0.735	0.831	0.763	0.862

Experimental setting Furthermore, we only keep the users who clicked at least once and the offers which were either shown or clicked by such users. For all interactions, we assigned +1 (positive feedback) if the user clicked on an offer that was shown to him, and -1 if the user did not click (negative feedback).

Finally, we sort the data w.r.t the timestamp and further divide it into 70% for training and 30% for testing, for all recommender algorithms. Such temporal split

makes more sense than random split because the interest of users change over time and is also more realistic with respect to the on-line setting. For experiments on KASANDR, we consider only the setting of *interacted items*.

Tables 6.7 and 6.8 reports MAP@5, 30 and 100 of all compared methods. As expected, non-machine learning methods namely Rand, Pop and PastI do not perform well. Similarly, we observe that MF also performs poorly when compared to FM and FFM. This result can be attributed to the fact that the number of new users in the test set is larger than the number of returning ones, and MF is well-known to fail to learn any latent factors for such users.

However, FM and its extension FFM are designed in a way that allow them to overcome this drawback and to learn from a reduced amount of positive feedback. For FFM, we include the *userId*, *offerId*, country code, offer category and merchant, as fields.

Then, we also propose to compute two supplementary count features from the raw data: the number of times the user clicked, regardless of the items, and the number of time an offer is clicked, regardless of the users. This version is referred to as FFM-F in the following. As shown in Table 6.8, FFM-F outperforms all the other models. We believe there is still room for improvement of FFM by doing such feature engineering; for instance by including the same count but computed on different time-windows, such as per week, as for now we consider the whole month.

One can also observe that results in terms of Macro MAP for FM and all its derivatives are usually higher than the results in terms of Micro MAP. A very simple explanation comes from the fact that the latter takes into account the size of the traffic of each country, and for instance, FFM-F obtains a MAP of 0.6397 for France versus a MAP of 0.9787 for Ireland which generates less traffic.

Finally, Table 6.9 reports the training and testing time for each approach on all countries. Not surprisingly, non-machine learning approaches are less computationally demanding. We can also see that FFM-F is only slightly slower than FFM, as it includes the two extra quantitative features but still much more faster than MF.

In this section, we presented KASANDR in a hope to encourage future research on recommendation systems using implicit feedback. It is designed to investigate a wide range of recommendation algorithms as it includes many contextual features about both customers and proposed offers. For comprehensiveness, we gave a description of

6. EXPERIMENTAL RESULTS

Table 6.9 – Training and testing time (in seconds).

	Rand	Pop	PastI	MF	FM	FFM	FFM-F
Train	341.759	630.112	139.409	36067.117	1142.096	1804.565	2179.745
Test	0	0	0	10259.487	444.924	462.800	490.498

side information and statistics. We also conducted experiments and compared strong baselines approaches, where we observed that, FFM was the best approach for this problem. We also demonstrated that feature engineering can greatly improve the results and should be more investigated on KASANDR.

Another interesting perspective include the integration of textual information available in KASANDR using the URL to retrieve the content of the page on which the item is presented, the tag associated to it, or the query string entered by the user for his search. For this purpose, models based on text mining, semantic analysis or natural language processing can be investigated. We also left aside other features in the experimentation such as the consumer’s behavior w.r.t. the type of device that s/he is using or the price of the items which we believe that they can greatly impact the performance of RS. In the next section, we discuss PANDOR, which has rich text information and suffers from popularity bias.

Results on PANDOR

Compared approaches First, we compare the performance of different state-of-the-art approaches that do not take into account the diversity for recommendation. The tested methods include: two non-machine learning approaches and five machine-learning based approaches which were developed to deal with highly sparse data and implicit feedback: Pop, Rand, Rank-ALS, BPR, FM, LightFM.

We don’t use Macro MAP@k or Micro MAP@k in case of PANDOR as we don’t have country information in PANDOR. However, as PANDOR suffers from popularity bias, we show diversity results by using EILD.

Evaluation Protocol We filter out users without a single click; the dataset contains 1,767,589 interactions from 119,536 unique users on 2,840 unique items. In addition, we sort all interactions according to time, then take the first 70% interactions for training the models, and the remaining 30% for testing. Finally, we consider both settings

w.r.t. to the set of items selected for the prediction as defined in section 6.2.

1. Item recommendation only relies on past *interacted offers*
2. The RS considers the *full set of items* as possible candidate for the prediction.

For the first setting, the average number of interacted items per user is 20.653 , i.e. the prediction is done over 20.653 items on average, while for the second one, the prediction is over 2840 items. The accuracy of the ranking list of items is evaluated by the Mean Average Precision (MAP) obtained for the set of top $k=1, 5$ and 10 items. Then, following [Wasilewski and Hurley, 2016a], we use the EILD (expected intra-list diversity) to measure diversity. High value of EILD indicates high diversity, and we report this metric at $k=10$. Because of the absence of meta information on the items, the distance between items is computed as the distance between their embeddings. We give more details about this choice on Page 94, where we give diversity results.

The results of comparing all methods on PANDOR, in both settings are summarized in Tables 6.10 and 6.11. One can see that on interacted items, LightFM significantly outperforms all competing approaches and achieved reasonable performance for this task. However, looking at the results of the second setting, the compared approaches give very low performance, and BPR-MF and NERvE give slightly better performance than LightFM and FM. Figure 6.3 provides a deeper analysis of these results for FM and LightFM, which are supposed to be particularly efficient for this type of data. This figure shows the rank of items as a function of their click-through rate (CTR) i.e. the ratio of clicks to impressions of an item, for FM, LightFM and Popularity. We can make two observations: (1) FM’s recommendation is driven by items with the highest CTR (in the top 1%); (2) LightFM behaves like Popularity, recommending only the most clicked items.

Next, we demonstrate how incorporating diversity using item embeddings, in Rank-ALS and NERvE, can enhance these results.

Diversity Results We run diversity tests on PANDOR. We propose to explore the ability of diversity in RS to overcome the strong bias induced by popular items, or items with high CTR. Also, we focus only on the setting in which we test on all items as most approaches fail to provide good results on such setting. To this end, we propose to

6. EXPERIMENTAL RESULTS

Table 6.10 – MAP@k obtained for all compared approaches on interacted items on PANDOR. The best results are in bold.

	MAP@1	MAP@5	MAP@10	EILD@10
Random	0.135	0.157	0.161	0.172
Popularity	0.249	0.262	0.266	0.080
FM (SGD)	0.244	0.269	0.273	0.191
BPR-MF	0.222	0.240	0.229	0.173
LightFM	0.479	0.526	0.535	0.099
NERVE _{c,p}	0.251	0.292	0.299	0.115
Rank-ALS	0.256	0.261	0.261	0.008

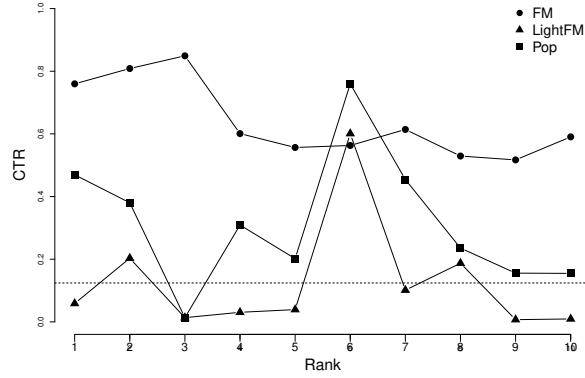


Figure 6.3 – Rank of recommended items as a function of their CTR. Here the results are for the setting where all items are considered for making prediction. The dot line represents the average CTR of all items.

Table 6.11 – MAP@k obtained for all compared approaches on all items on PANDOR.

	MAP@1	MAP@5	MAP@10	EILD@10
Random	9.934e-05	0.0001	0.0001	0.536
Popularity	0.007	0.009	0.011	0.396
FM (SGD)	0.001	0.002	0.003	0.534
BPR-MF	0.005	0.008	0.010	0.493
LightFM	0.0002	0.0008	0.002	0.287
NERvE _{c,p}	0.006	0.008	0.010	0.560
Rank-ALS	0.002	0.002	0.003	0.564

Table 6.12 – Results of NERvEcoupled with diversity. HM denotes the harmonic mean of MAP and EILD.

Metric Maximized	β	MAP@10	EILD
MAP@10	0.0001	0.010	0.633
EILD	0.1	0.001	0.666
HM(MAP@10,EILD)	0.0001	0.010	0.633
HM(MAP@10,EILD) while maximizing diversity	-0.75	0.006	0.635

evaluate two approaches. The first one was initially proposed by [Wasilewski and Hurley, 2016a] and consider the objective function of Rank-ALS [Takács and Tikk, 2012] augmented with a regularization term that consists of the intra-list diversity (ILD) measure. Then, without loss of generality, we propose to build upon NERvE. The diversity regularizers, we add here for RankALS or NERvE, can be used with any loss function. In [Wasilewski and Hurley, 2016a], the authors used the movies’ genre to compute distances between two items. We propose to compute item embeddings as meta data [Barkan and Koenigstein, 2016a]. Here, we would like to stress on the fact that computing embeddings with the Item2Vec [Barkan and Koenigstein, 2016a] technique to measure diversity is a fresh departure from previous works on this topic; indeed, in our case, item diversity is not related to the characteristics of the items themselves, such as the genre, or the category, but rather to the diversity of the sequence of items displayed to users. This means that our goal is to, somehow, force the RS algorithm to display various diverse sequences of items to each user. We compute item embeddings, with Gensims based Skip-Gram implementation of Word2Vec (adapted to Item2Vec). We set the dimension to 20 and consider 3 as the context window.

6. EXPERIMENTAL RESULTS

Table 6.13 – Results of RankALS coupled with diversity. HM denotes the harmonic mean of MAP and EILD

Metric Maximized	<i>regularizer</i>	MAP@10	EILD
MAP@10	PLapDQ-min	0.018	0.552
EILD	No-Regularizer	0.0002	0.692
HM(MAP@10,EILD)	PLapDQ-min	0.018	0.552
HM(MAP@10,EILD) while maximizing diversity	DQ-max	0.016	0.553

Table 6.14 – By Introducing diversity we are able to increase both relevance of the items and diversity of items

	Before Diversity				After Diversity			
	MAP@1	MAP@5	MAP@10	EILD@10	MAP@1	MAP@5	MAP@10	EILD@10
NERvE	0.006	0.008	0.010	0.561	0.009	0.009	0.010	0.633
RankALS	0.002	0.002	0.003	0.564	0.010	0.014	0.016	0.553

NERvE and RankALS with diversity In Section 5.4, we described the way of incorporating diversity in NERvE. Next, we describe the way of doing it in RankALS and the results we obtained after doing the same. In RankALS [Wasilewski and Hurley, 2016a], a diversity regularization term is added, thus taking into account diversity in a single step learning, as we propose for NERvE. From the EILD metric, the authors derived various forms for the regularization term, all based on a distance matrix between items using some available characteristics. In this work, we compute the distance between items embeddings as described previously.

Best results are summarized in Tables 6.12, 6.13 and 6.14. Overall, one can observe that in both cases, adding diversity based on embeddings, results in significant boost of the RS performance in terms of MAP, and allows Rank-ALS and NERvE to outperform BPR-MF (which was found to be the strongest baseline in this setting). For NERvE, one can also note that by taking negative β , we are actually able to improve MAP and EILD computed in Table 6.11. This observation stresses the fact that by introducing more diversity in recommendations on data sets such as PANDOR, which were built by popularity biased algorithms, we are actually able to improve the relevance of recommended offers. For Rank-ALS, the gap in terms of MAP between the versions with and without diversity is even more important.

Topic-Modelling application to RS

In chapter 4, we had first introduced topic models and how topics derived from such topic models can be used to make recommendations. In this section, we use one of the topic models described in Section 4.3, namely TM-LDA, and feed the topics derived from it, as contextual information to Factorization Machines.

Experimental setting We first sort the dataset temporally. Then, we remove all the users who did not do a single click during the whole time period. We, then, take first 80% for training and remaining 20% for testing. We make use of Page Text for running TM-LDA. This is because pages, which users are browsing on, depict the interest of the user and indeed, offers to be shown at any given time should match the interest of the user. We remove all the pages from test period which were not there in the training period. Various statistics of data are summarized in Table 6.15.

Table 6.15 – Overall Dataset Aggregate Statistics.

#Unique Pages	1770
# Total Interactions with Page Text	605,386
# Interactions in training data	578618
# Interactions in test data	26768

As for the baselines, we compare against Factorization Machine without any contextual information. We also compare against two non-machine learning baselines, namely, Random and Popularity. The results are shown in Table 6.16. Using TM-LDA based topics as contextual information in Factorization Machines improves recommendations as shown in Table 6.16.

Table 6.16 – MAP@k improves after putting TM-LDA-based topics as contextual information in Factorization Machines on interacted items on PANDOR. The best results are in bold.

	MAP@1	MAP@5	MAP@10
Random	0.135	0.157	0.161
Popularity	0.249	0.262	0.266
Factorization Machines (FM)	0.244	0.269	0.273
TM-LDA-Based FM using Page Text	0.385	0.390	0.389

6.5 Conclusion

In this chapter, we first described the results on NERvE, a neural network to learn good representation and pairwise ranking function simultaneously. We assessed and validated the proposed approach through extensive experiments, using four popular collections proposed for the task of recommendation. Furthermore, we studied two different settings for the prediction phase and demonstrate that the performance of each approach is strongly impacted by the set of items considered for making the prediction. In both the settings, NERvE outperforms the other approaches using MAP as a metric. We run recommender baselines meant to learn from implicit feedback on KASANDR and set benchmarks which may be helpful for RS community. We then describe PANDOR and baseline results on PANDOR and how the preliminary results suffer from popularity bias, a known problem in RS. We introduce diversity in loss functions of NERvE and Rank-ALS, and depict improvement in preliminary results. Finally, we showcase the use of contextual information such as *time* and *text* by using TM-LDA based topics in Factorization Machines and show using this information can lead to significant improvement of results.

Chapter 7

Conclusions and future perspectives

In this thesis, we first presented and focused on Kelkoo's June data and Purch's one month data of implicit preference signals (clicks) from twenty European countries. Working with industry data presents interesting challenges. Kelkoo's data was big enough (353 GB compressed and 650 GB uncompressed) to not fit in one system. This led us to spend considerable time on reading the data in main memory and preprocess and cleaning the data so that RS models could be build on the data. SPARK, which is a technology developed for handling big data and building machine learning models in a distributed manner, was used to do pre-processing and build dataset formats on which RS baselines can be built.

There were numerous bugs found in the initial stages of cleaning and preprocessing the data. For instance, we found that maximum number of the clicks were done by bots and not a human and many users have done no click at all. We also found that many offers which were being clicked were never shown to the users. Additionally, users were tracked by maintaining cookies and this user-tracking system was not profound.

Having removed/minimized the effect of the bugs, we started reading literature and found out that all the traditional classification based RS models use ratings as input and converting clicks (which were available to us) into ratings was an unnatural thing to do. Nevertheless, we started by treating number-of-clicks as a rating and no-click as zeros. This led to a problem of mega-sparsity where almost all interactions were no-clicks.

We started building models on this big and skewed data. We began with simple approaches such as recommending most popular items and items, which user interacted in past. Our next objective was to see if Machine Learning based approaches can do

better than these simple approaches, which served as the baselines which needed to be outperformed. We saw that Matrix Factorization based approach, which was the winning approach for Netflix prize, did not perform very well on this data. Most probable explanation for this in literature has been (and which we also observed) that matrix factorization based approach tries to optimize Root Mean Squared Error (RMSE) which is well suited for ratings and not clicks. Additionally, matrix factorization based approaches do not make use of contextual information. Some of these shortcomings were overcome by making use of Factorization Machines and Field-Aware Factorization Machines. We tried to put FFM in Kelkoo's production system and is one of our future perspectives. We also built numerous RS models developed for implicit feedback for KASANDR and compared their performances. KASANDR dataset contains twenty countries and we evaluate the RS baselines on all the twenty countries using Micro-MAP and Macro-MAP.

Having computed the basic statistics and run the baselines, we contributed KASANDR and the baseline results, as well as rich meta-information accompanying it as a benchmark, which can be useful for research community working in RS. We presented and described these contextual features present in the data set.

Then, we started working on another industry dataset, coined the name, PANDOR. We computed all the basic statistics and did feature study as before. We also noted that PANDOR is the dataset in which items recommended to the user have popularity bias in it. Due to this bias, performance of all the RS baselines get affected. In order to overcome the popularity bias, we introduce *diversity term* in cost function of two ranking algorithms. We noted that by introducing diversity, performance of baselines improved. As before, we contributed this dataset along with contextual information accompanying it to RS community. Diversity results were also presented along with other learning-to-rank based baselines developed for handling implicit feedback.

When optimizing on clicks, we came to understand that main objective is to come up with an item, which user is most likely to click, so that this item can be presented on top of the list of recommended items. We also observed that deep learning based methods are increasingly being applied to the problem of RS leading to considerable increase in performance over traditional methods. This observation led to switch of our focus from classification based approaches to learning-to-rank and neural network based methods.

7. CONCLUSIONS AND FUTURE PERSPECTIVES

We developed a *neural network*, coined the term NERVE, which learns by minimizing two losses simultaneously during backpropagation. One of the losses involved focuses on learning representations. Representations are based on a technique developed recently called *Item2Vec*. The other loss function stresses on *pairwise learning-to-rank function*. The Neural Network framework was developed in Tensorflow. Tensorflow implements all the backpropagation based gradients by itself. We just need to specify the structure of the network and associated loss functions. We rigorously tuned parameters involved in learning the predictions and monitor the performance of this neural network by trying out many parameter values for all the parameters. We study the performance of this neural network under three different settings. Two settings consist of minimizing the individual loss functions and the third setting consists of minimizing both the losses simultaneously. We test this model on many RS data sets, including KASANDR. We noted that by minimizing these two losses simultaneously, we are able to outperform many popular baselines on many implicit feedback datasets.

New to PANDOR dataset was also the rich textual information accompanying it. As KASANDR, PANDOR also suffers from the problem of mega-sparsity as most of the interactions are no-clicks (non-positive feedback). In order to handle the problem of sparsity, many works have suggested to make use of meta-data. We thought to make use of topic outputs of time-aware topic models to be used in RS models as contextual information. So, we used two novel *time-aware topic modelling* techniques, namely, TM-ATAM and T-ATAM. Both of these topic modelling techniques are time-aware and are fresh departure from topic modeling techniques which are time-oblivious. TM-ATAM involves post-processing of inferences in order to come up with a transition matrix, entries of which, model the quantity by which topics will transform into another with time. In particular, it solves a least squares problem between distributions of consecutive temporal posts. T-ATAM treats time as a random observed variable inside the model itself and doesn't require any post processing. We show that these time-aware topic models perform very well on health dataset by using perplexity. Then, we test our hypothesis that, using contextual information can help RS models. In order to do that, we show that topics learned by using a time-aware general purpose model (such as TM-LDA) improves performance of Factorization Machines for recommending products. With that said, we list down some of the ideas pertaining to temporal topic models we would like to test as future perspectives in next section.

In all the above mentioned RS models, one perennial problem facing RS is that, we test against the offers shown to the user in the test set. However, offers shown to user are the result of the model, which was used to show those offers. But, there could have been many offers which user would have clicked had they been shown to him/her. We discuss this problem of bias introduced because of using a particular model in next section and approaches which have been suggested to overcome this problem.

Future perspectives

Here, we made an attempt to solve some of the central problems surrounding RS. However, numerous improvements and extensions over the models which we developed and we aspire to develop are possible. In this vein, we provide different perspectives in which the existing models can be extended or new models can be developed.

First of all, the idea of feeding contextual information present in `NERvE`, and looking at the change in effects in performance excites us. This contextual information can either be item meta-information present in the data or topics inferred from time-aware topic models. We also intend to apply topic modeling techniques to content-filtering recommendation techniques. We would like to extend `NERvE` in order to take into account additional contextual information regarding users and/or items. More specifically, we are interested in the integration of data of different natures, such as text or demographic information. We believe that this information can be taken into account without much effort and by doing so, it is possible to improve the performance of our approach and tackle the problem of providing recommendation for new users/items at the same time, also known as the cold-start problem. The second important extension will be the development of an on-line version of the proposed algorithm in order to make the approach suitable for real-time applications and on-line advertising.

A natural problem arising from the online aspect of RS, and commonly referred to as cold-start, describes a situation when one needs to address new users and/or items. Traditional RS rely on past interaction data in order to generate new recommendations. Therefore, those approaches fail to generate relevant recommendations for new users and items arriving into the system due to missing information about their past interactions. The simple strategy is to recommend the most popular items to a new user.

7. CONCLUSIONS AND FUTURE PERSPECTIVES

Another way is to use, when available, contextual information regarding the new user, such as the gender or demographic information (i.e. country of residence), to recommend items liked by similar users for whom we have past feedback [Balabanovic and Shoham, 1997; Basu et al., 1998; Claypool et al., 1999; Pazzani, 1999]. To address the issue of item cold-start, one can use content-based approaches, relying on the use of contextual information such as the title of the offer or the genre of a movie to infer similarity with other items already present in the system [Chu and Park, 2009; Good et al., 1999; Park et al., 2006; Schein et al., 2002; Stern et al., 2009]. This idea of using contextual information excites us and we would like to extend NERVE by using contextual information in order to handle the problem of user cold-start and item cold-start.

Treating recommender problem as a sequential learning problem is interesting from many view-points. Problem of recommendations is a natural sequential learning problem because user’s interest changes rapidly and recommender model also need to adapt to ever-changing user’s interests. In this vein, collaborative bandits (reinforcement learning) can be applied to come up with better recommendations. Collaborative bandits handle *explore-exploit dilemma* very well, where, explore step tries to discover new interests of the user and meet ever-changing demands of the user, while exploit step tries to recommend to the user using whatever has been learned about the user so far. Also, user can click on what we show. But, what we show is the result of what our model predicted was good. In other words, there are no counterfactuals. This is the problem with implicit feedback, that, it has got no real negatives. *Explore-exploit approaches* are potential solution to this problem.

Another line of research, we want to consider, is intent-aware or session-aware recommendations. *Recurrent Neural Networks* (RNNs) also handle recommendations as a sequence and have been shown to perform very well recently in session based recommendations. We would like to also adapt the techniques we have developed to adapt to the framework of RNNs and monitor the performance.

One more future perspective, which can have impact in performance of RS, that we want to consider is *value-aware recommendations*. The approaches developed in this domain consider that not all clicks/actions have same “reward”. These approaches focus on the trade-offs between long-term retention vs. short-term clicks (clickbait). In this thesis, we have been focussing on *clicks* as target, but focussing just on clicks

may not generate revenue for the company or value for the customer.

One area of research is in the direction of full-page optimization. This area takes recommendations into *multi-task learning* domain, where we would like to *jointly* optimize for the set of items to recommend and their placement on the page and also incorporate those other factors such as diversity, freshness, coverage of items being recommended.

Personalizing *how* we recommend and not just *what* we recommend is one another line of research we would like to explore. Ideal balance of diversity, novelty, popularity, freshness, etc. may depend on the person. In other words, how we present items or explain recommendations can also be personalized. These techniques also aim to balance the needs of lean-back users and power users on the interaction level.

All in all, a lot of topics and challenging problems still remain unsolved in RS and recommender problem, in general, is still far from being perfectly solved problem.

Published articles

1. Learning to Recommend Diverse Items over Implicit Feedback on PANDOR with Charlotte Laclau, Massih-Reza Amini, RecSys 2018.
2. Health Monitoring on Social Media over Time with Sihem Amer-Yahia, Marianne Clausel, Majdeddine Rebai, Son T Mai, Massih-Reza Amini in IEEE Transactions on Knowledge and Data Engineering Journal, 30(8), pp. 1467–1480, 2018.
3. KASANDR: A Large-Scale Dataset with Implicit Feedback for Recommendation with Charlotte Laclau, Massih-Reza Amini, Gilles Vandelle, Andr Bois-Crettez, SIGIR 2017.
4. Health Monitoring on Social Media over Time with Shashwat Mishra, Sihem Amer-Yahia, Marianne Clausel, Massih-Reza Amini, SIGIR 2016.

Submitted articles

5. Representation Learning and Pairwise Ranking for Implicit Feedback in Recommendation Systems with Mikhail Trofimov, Oleg Horodnitskii, Charlotte Laclau, Yury Maximov, Massih-Reza Amini

7. CONCLUSIONS AND FUTURE PERSPECTIVES

References

- Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of RecSys*, pages 42–46, New York, NY, USA, 2017. ACM. 6, 76
- Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. In *Proceedings of COLT*, pages 87–98, (2008). 75
- Massih-Reza Amini and Nicolas Usunier. *Learning with Partially Labeled and Inter-dependent Data*. Springer, New York, NY, USA, 2015. ISBN 978-3-319-15726-9. 66
- Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006. 48
- Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003. doi: 10.1023/A:1020281327116. URL <https://doi.org/10.1023/A:1020281327116>. 52
- Marko Balabanovic and Yoav Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997. 105
- Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *WSDM*, 2012. 23
- Trapit Bansal, David Belanger, and Andrew McCallum. *Ask the GRU*: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 107–114, 2016. 24

- Oren Barkan and Noam Koenigstein. ITEM2VEC: neural item embedding for collaborative filtering. In *International Workshop on Machine Learning for Signal Processing, MLSP*, pages 1–6, 2016a. [77](#), [96](#)
- Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. In *Proceedings of the Poster Track of RecSys*, 2016b. [27](#)
- Justin Basilico and Yves Raimond. Déjà vu: The importance of time and causality in recommender systems. In *Proceedings of RecSys*, page 342. ACM, 2017. [31](#), [50](#)
- Chumki Basu, Haym Hirsh, and William W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720. AAAI Press / The MIT Press, 1998. [105](#)
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. [26](#)
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning*, 3:993–1022, 2003. [51](#), [52](#)
- Rubi Boim, Tova Milo, and Slava Novgorodov. Diversification and refinement in collaborative filtering recommender. In *Proceedings of CIKM*, pages 739–744, 2011. [28](#)
- Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 421–436. 2012. [74](#)
- Keith Bradley and Barry Smyth. Improving recommendation diversity, 2001. [6](#), [28](#)
- Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010. [23](#)
- Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML*, pages 89–96, 2005. [24](#)

- Pedro G. Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.*, 24(1-2):67–119, 2014. [50](#)
- Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR*, pages 335–336, 1998. [28](#)
- Rich Caruana, Shumeet Baluja, and Tom M. Mitchell. Using the future to sort out the present: Rankprop and multitask learning for medical risk evaluation. In *Proceedings of NIPS*, pages 959–965, 1995. [24](#)
- Pablo Castells, Neil J. Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 881–918. 2015. [34](#)
- Sotirios Chatzis, Panayiotis Christodoulou, and Andreas S. Andreou. Recurrent latent variable networks for session-based recommendation. *CoRR*, abs/1706.04026, 2017. [24](#), [50](#)
- Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhiming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *Proceedings of NIPS*, pages 315–323. Curran Associates, Inc., 2009. [22](#)
- Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. Learning to recommend accurate and diverse items. In *Proceedings of the 26th International Conference on World Wide Web, WWW ’17*, 2017. [28](#)
- Soudip Roy Chowdhury, Muhammad Imran, Muhammad Rizwan Asghar, Sihem Amer-Yahia, and Carlos Castillo. Tweet4act: Using incident-specific profiles for classifying crisis-related messages. In *10th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Baden-Baden, Germany, May 12-15, 2013.*, 2013. [58](#)
- Wei Chu and Seung-Taek Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of WWW*, pages 691–700. ACM, 2009. [105](#)

- Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper, 1999. 105
- William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999. 21
- Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1007/BF00994018. URL <http://dx.doi.org/10.1007/BF00994018>. 60
- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys ’16, 2016a. 24
- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of RecSys*, pages 191–198, 2016b. 27
- Koby Crammer and Yoram Singer. Pranking with ranking. In *Proceedings of NIPS*, pages 641–647, 2001. 20
- Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Recurrent coevolutionary feature embedding processes for recommendation. *CoRR*, abs/1609.03675, 2016. 24
- Thomas Davidson, Dana Warmley, Michael W. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, Québec, Canada, May 15-18, 2017.*, pages 512–515, 2017. 58
- Thomas Deselaers, Tobias Gass, Philippe Dreuw, and Hermann Ney. Jointly optimising relevance and diversity in image retrieval. In *Proceedings of CIVR*, 2009. 28
- Mukund Deshpande and George Karypis. Item-based top- N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004. 13

- Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05*, 2005. 50
- Marina Drosou and Evaggelia Pitoura. Diversity over continuous data. *IEEE Data Eng. Bull.*, 32(4):49–56, 2009. 28
- Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of WWW*, pages 278–288, 2015. 24
- Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4: 933–969, 2003. 21
- Simon Funk. Netflix update: Try this at home. <http://sifter.org/simon/journal/20061211.html>, 2006. 15
- Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. Offline A/B testing for recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 5-9, 2018*, pages 198–206, 2018. 35
- Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Badrul Munir Sarwar, Jonathan L. Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446. AAAI Press / The MIT Press, 1999. 105
- Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of SIGKDD*, pages 1809–1818, 2015. 27
- Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. *CoRR*, abs/1606.07154, 2016. 25, 27

- Élie Guàrdia-Sebaoun, Vincent Guigue, and Patrick Gallinari. Latent trajectory modeling: A light and efficient way to introduce time in recommender systems. In *Proceedings of RecSys*, pages 281–284, 2015. [27](#)
- Frédéric Guillou. *On Recommendation Systems in a Sequential Context. (Des Systèmes de Recommandation dans un Contexte Séquentiel)*. PhD thesis, Charles de Gaulle University, Lille, France, 2016. [24](#), [30](#)
- Asela Gunawardana and Guy Shani. Evaluating recommender systems. In *Recommender Systems Handbook*, pages 265–308. 2015. [30](#), [35](#)
- F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015. [83](#)
- Ruining He and Julian McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. *CoRR*, abs/1510.01784, 2015. [24](#)
- Ruining He and Julian McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 144–150, 2016. [24](#)
- Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of SIGIR*, pages 549–558. ACM, 2016. [5](#)
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of WWW*, pages 173–182, 2017. [27](#)
- Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 230–237, 1999. [12](#)
- Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1): 5–53, 2004. [6](#), [28](#)

- Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *CoRR*, abs/1706.03847, 2017. [24](#), [50](#)
- Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2015. [24](#), [50](#)
- Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 241–248, 2016. [24](#), [50](#)
- Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*, pages 263–272, 2008a. [16](#), [81](#)
- Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM*, pages 263–272, 2008b. [81](#)
- Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings ICDM*, pages 263–272, 2008c. [17](#)
- Neil J. Hurley. Personalised ranking with diversity. In *Proceedings of RecSys*, pages 379–382, 2013. [29](#)
- Michael Jahrer and Andreas Tscher. Collaborative filtering ensemble for ranking. In *KDD Cup*, volume 18 of *JMLR Proceedings*, pages 153–167. JMLR.org, 2012. [23](#)
- Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Symposium on Theory of Computing Conference*, pages 665–674, 2013. [16](#)
- S. Janson. Large Deviations for Sums of Partly Dependent Random Variables. *Random Structures and Algorithms*, 24(3):234–248, 2004. [66](#), [67](#)
- Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of SIGKDD*, pages 133–142, 2002. [21](#)

- Thorsten Joachims and Adith Swaminathan. Counterfactual evaluation and learning for search, recommendation and ad placement. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR*, pages 1199–1201, 2016. [35](#)
- Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for CTR prediction. In *Proceedings of RecSys*, pages 43–50, 2016a. [81](#)
- Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50, 2016b. [19](#)
- Yuchin Juan, Damien Lefortier, and Olivier Chapelle. Field-aware factorization machines in a real-world online advertising system. In *Proceedings of WWW (Companion Volume)*, pages 680–688. ACM, 2017. [19](#)
- Tomonari Kamba, Krishna Bharat, and Michael C. Albers. The krakatoa chronicle: An interactive personalized newspaper on the web. *World Wide Web Journal*, 1(1), 1996. [11](#)
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. [87](#)
- Ron Kohavi. Online controlled experiments: Lessons from running a/b/n tests for 12 years. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, 2015. [35](#)
- Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 426–434, 2008. [16](#), [17](#)
- Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010. [50](#)
- Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009. [81](#)

- Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with RecSys.*, pages 14–21, 2015. [81](#), [87](#)
- Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web, WWW '14*, 2014. [24](#)
- E.L. Lehmann and H.J.M. D'Abrera. *Nonparametrics: statistical methods based on ranks*. Springer, 2006. [88](#)
- O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, pages 2177–2185, 2014. [26](#)
- Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Proceedings of NIPS*, pages 897–904, 2007. [20](#)
- Xiaohui Li and Tomohiro Murata. Using multidimensional clustering based collaborative filtering approach improving recommendation diversity. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Macau, China, December 4-7, 2012*, pages 169–174, 2012. [28](#)
- Dawen Liang, Jaan Allosa, Laurent Charlin, and David M. Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of RecSys*, pages 59–66, 2016. [27](#), [81](#)
- Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. [20](#)
- Xin Liu and Karl Aberer. Towards a dynamic top-n recommendation framework. In *Eighth ACM Conference on Recommender Systems, RecSys '14*, pages 217–224, 2014. [24](#)
- Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011. [11](#)

- Widad Machmouchi and Georg Buscher. Principles for the design of online A/B metrics. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 589–590, 2016. 35
- Lydia Manikonda and Munmun De Choudhury. Modeling and understanding visual attributes of mental health disclosures in social media. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017.*, pages 170–181, 2017. 58
- Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, pages 43–52, 2015. 24
- C. McDiarmid. On the method of bounded differences. *Survey in Combinatorics*, pages 148–188, 1989. 67
- Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *Extended Abstracts Proceedings of the Conference on Human Factors in Computing Systems, CHI*, pages 1097–1101, 2006. 6, 28
- David McSherry. Diversity-conscious retrieval. In *Advances in Case-Based Reasoning, 6th European Conference, ECCBR*, pages 219–233, 2002. 6, 28
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013a. URL <http://arxiv.org/abs/1301.3781>. 26
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013b. 26
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119. 2013c. 26

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, 2013d. [26](#)
- Raymond J. Mooney and Lorie Roy. Content-based book recommending using learning for text categorization. *CoRR*, cs.DL/9902011, 1999. [11](#)
- Cataldo Musto, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. Learning word embeddings from wikipedia for content-based recommender systems. In *Proceedings of ECIR*, volume 9626 of *Lecture Notes in Computer Science*, pages 729–734. Springer, 2016. [50](#)
- Cataldo Musto, Marco de Gemmis, Giovanni Semeraro, and Pasquale Lops. A multi-criteria recommender system exploiting aspect-based sentiment analysis of users’ reviews. In *Proceedings of RecSys*, RecSys ’17, 2017. [50](#)
- Thomas Nadelec, Elena Smirnova, and Flavian Vasile. Specializing joint representations for the task of product recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys*, pages 10–18, 2017. [25](#), [27](#)
- Seung-Taek Park, David M. Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. In *Proceedings of SIGKDD*, pages 699–705, 2006. [105](#)
- Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of RecSys*, pages 11–18, New York, NY, USA, 2008. ACM. [48](#)
- Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. 01 2007. [16](#)
- Bibek Paudel, Thilo Haas, and Abraham Bernstein. Fewer flops at the top: Accuracy, diversity, and regularization in two-class collaborative filtering. In *Proceedings of RecSys*, pages 1–6. ACM, 2017. [6](#)
- Michael J. Paul and Mark Dredze. You Are What You Tweet: Analyzing Twitter for Public Health. In *ICWSM’11*, 2011. [61](#)

- Michael J. Paul and Roxana Girju. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *AAAI*. AAAI Press, 2010. [53](#), [57](#)
- Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 1999. [105](#)
- Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007. [11](#)
- Jeffrey Pennington, Felix X. Yu, and Sanjiv Kumar. Spherical random features for polynomial kernels. In *Proceedings of NIPS*, pages 1846–1854, 2015. [26](#)
- Jean-François Pessiot, Tuong-Vinh Truong, Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. Learning to rank for collaborative filtering. In *Proceedings of ICEIS*, pages 145–151, 2007. [21](#)
- Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. Product-based neural networks for user response prediction. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, pages 1149–1154, 2016. [24](#)
- Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 130–137, 2017. [24](#), [50](#)
- Liva Ralaivola and Massih-Reza Amini. In *Proceedings of ICML*, pages 2436–2444, 2015. [68](#), [69](#)
- Steffen Rendle. Factorization machines. In *Proceedings of ICDM*, pages 995–1000, 2010. [18](#), [81](#)
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *Proceedings of UAI*, pages 452–461, 2009. [22](#), [71](#), [81](#)

- Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 811–820, 2010. [50](#)
- Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of SIGIR*, pages 635–644. ACM, 2011. [19](#)
- Marco Túlio Ribeiro, Anísio Lacerda, Adriano Veloso, and Nivio Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of RecSys*, pages 19–26, 2012. [29](#)
- Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Monica Bianchini. A neural network approach for learning object ranking. In *Proceedings of ICANN*, pages 899–908, 2008. [24](#)
- Leonardo Rigutini, Tiziano Papini, Marco Maggini, and Franco Scarselli. Sortnet: Learning to rank by a neural preference function. *IEEE Trans. Neural Networks*, 22(9):1368–1380, 2011. [24](#)
- Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-session modeling for session-based recommendation. In *Proceedings of the 2Nd Workshop on Deep Learning for Recommender Systems, DLRS 2017, 2017*. [24](#), [50](#)
- Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of ICML*, pages 791–798, 2007. [24](#)
- Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000. [15](#)
- Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295, 2001. [13](#)

- Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of SIGIR*, pages 253–260. ACM, 2002. [105](#)
- Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005. [50](#)
- John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972. [70](#)
- Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. Swivel: Improving embeddings by noticing what’s missing. *arXiv preprint arXiv:1602.02215*, 2016. [26](#)
- Lei Shi. Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach. In *Proceedings of RecSys*, pages 57–64, 2013. [28](#)
- Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of RecSys*, pages 269–272, 2010. [21](#)
- Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Sixth ACM Conference on Recommender Systems, RecSys*, pages 139–146, 2012. [21](#)
- Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In *Seventh ACM Conference on Recommender Systems, RecSys*, pages 431–434, 2013. [21](#)
- Sumit Sidana, Shashwat Mishra, Sihem Amer-Yahia, Marianne Clausel, and Massih-Reza Amini. Health monitoring on social media over time. In *SIGIR*, pages 849–852. ACM, 2016. [51](#), [59](#)
- Sumit Sidana, Charlotte Laclau, Massih-Reza Amini, Gilles Vandelle, and André Bois-Crettez. KASANDR: A large-scale dataset with implicit feedback for recommenda-

- tion. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1245–1248, 2017. [38](#), [83](#)
- Sumit Sidana, Sihem Amer-Yahia, Marianne Clausel, Majdeddine Rebai, Son T. Mai, and Massih-Reza Amini. Health monitoring on social media over time. *IEEE Trans. Knowl. Data Eng.*, 30(8):1467–1480, 2018a. [51](#), [59](#)
- Sumit Sidana, Charlotte Laclau, and Massih-Reza Amini. Learning to recommend diverse items over implicit feedback on PANDOR. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 427–431, 2018b. [38](#)
- Elena Smirnova and Flavian Vasile. Contextual sequence modeling for recommendation with recurrent neural networks. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2017, Como, Italy, August 27, 2017*, pages 2–9, 2017. [24](#), [50](#)
- Barry Smyth and Paul McClave. Similarity vs. diversity. In *Proceedings of International Conference on Case-Based Reasoning, ICCBR*, pages 347–361, 2001. [6](#), [28](#)
- David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of WWW*, pages 111–120. ACM, 2009. [105](#)
- Ruilong Su, Li’ang Yin, Kailong Chen, and Yong Yu. Set-oriented personalized ranking for diversified top-n recommendation. In *Proceedings of RecSys*, pages 415–418, 2013. [29](#)
- Alessandro Suglia, Claudio Greco, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. A deep architecture for content-based recommendations exploiting recurrent neural networks. In *Proceedings of UMAP*, pages 202–211. ACM, 2017. [24](#)
- Gábor Takács and Domonkos Tikk. Alternating least squares for personalized ranking. In *Proceedings of RecSys*, pages 83–90, 2012. [23](#), [76](#), [81](#), [96](#)

- Gábor Takács, István Pilászy, Botyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *SIGKDD Explorations*, 9(2):80–83, 2007. [16](#)
- Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. *CoRR*, abs/1606.08117, 2016. [24](#), [50](#)
- Liang Tang, Yexi Jiang, Lei Li, and Tao Li. Ensemble contextual bandits for personalized recommendation. In *RecSys*, pages 73–80. ACM, 2014. [30](#)
- Bartłomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 273–276, 2016. [24](#), [50](#)
- Nicolas Usunier, Massih Amini, and Patrick Gallinari. A data-dependent generalisation error bound for the AUC. In *ICML workshop on ROC Analysis in Machine Learning*, 2005. [65](#)
- Nicolas Usunier, Massih-Reza Amini, and Patrick Gallinari. Generalization error bounds for classifiers trained with interdependent data. In *Proceedings of NIPS*, pages 1369–1376, 2006. [67](#), [68](#)
- Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 2643–2651, 2013. [24](#)
- Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2000. [71](#)
- Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of RecSys*, pages 225–232, 2016a. [27](#)

- Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of RecSys*, pages 225–232, 2016b. [27](#)
- Maksims Volkovs and Guang Wei Yu. Effective latent models for binary feedback in recommender systems. In *Proceedings of SIGIR*, pages 313–322, 2015. [71](#)
- Maksims Volkovs and Richard S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*, pages 2303–2311, 2012. [23](#)
- Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. *CoRR*, abs/1409.2944, 2014. [24](#)
- Hao Wang, Xingjian Shi, and Dit-Yan Yeung. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. *CoRR*, abs/1611.00454, 2016. [24](#)
- Yu Wang, Eugene Agichtein, and Michele Benzi. TM-LDA: Efficient Online Modeling of Latent Topic Transitions in Social Media. In *KDD’12*, pages 123–131, 2012. [55](#)
- Jacek Wasilewski and Neil Hurley. Incorporating diversity in a learning to rank recommender system. In *Proceedings of FLAIRS*, pages 572–578, 2016a. [29](#), [76](#), [94](#), [96](#), [97](#)
- Jacek Wasilewski and Neil Hurley. Intent-aware diversification using a constrained PLSA. In *Proceedings of RecSys*, pages 39–42, 2016b. [29](#)
- Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alexander J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems*, pages 1593–1600, 2007. [21](#)
- Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the International Conference on Web Search and Data Mining*, pages 153–162, 2016. [24](#)
- Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *ISMIR*, 2014. [30](#)

- Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of SIGIR*, pages 391–398, 2007. [21](#)
- Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of SIGIR*, pages 107–114, 2008. [21](#)
- Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of RecSys*, pages 123–130, 2008. [6](#), [28](#)
- Mi Zhang and Neil Hurley. Novel item recommendation by user profile partitioning. In *2009 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2009, Milan, Italy, 15-18 September 2009, Main Conference Proceedings*, pages 508–515, 2009. [28](#)
- Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, CIKM '13*, 2013. [30](#)
- Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW*, pages 167–176, 2018. [30](#)
- Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. A neural autoregressive approach to collaborative filtering. *CoRR*, abs/1605.09477, 2016. [24](#)
- Yunhong Zhou, Dennis M. Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management, 4th International Conference, AAIM*, pages 337–348, 2008. [16](#)
- Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of WWW*, pages 22–32, 2005. [28](#)
- Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. Using temporal data for making recommendations. *CoRR*, abs/1301.2320, 2013. [50](#)

