Multi-view Latent Factor Models for Recommender Systems

Da Costa, Felipe Soares

DOI (link to publication from Publisher):
10.54337/aau306504743

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Da Costa, F. S. (2019). *Multi-view Latent Factor Models for Recommender Systems*. Aalborg Universitetsforlag. https://doi.org/10.54337/aau306504743

MULTI-VIEW LATENT FACTOR MODELS FOR RECOMMENDER SYSTEMS

BY
FELIPE SOARES DA COSTA

DISSERTATION SUBMITTED 2019

AALBORG UNIVERSITY
DENMARK

Multi-view Latent Factor Models for Recommender Systems

Ph.D. Dissertation

Felipe Soares da Costa

Dissertation submitted in February, 2019

Dissertation submitted:	January, 2019
PhD supervisor:	Associate Professor Peter Dolog Aalborg University
PhD committee:	Associate Professor Hua Lu (chairman) Aalborg University Professor Dietmar Jannach University of Klagenfurt Professor Judith Masthoff Utrecht University
PhD Series:	Technical Faculty of IT and Design, Aalborg University
Department:	Department of Computer Science

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-391-4

Published by:
Aalborg University Press
Langagervej 2
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2019

Abstract

The goal of recommender systems is to predict the most preferred items for a user according to her/his preferences. The most popular technique of recommender systems is collaborative filtering, which is commonly classified as memory- or model-based. The model-based technique aims to identify hidden features through machine learning methods. Among model-based approaches, latent factor models became popular due to its high accuracy in predicting user's preferences. However, the latent factor models suffer from sparsity, lack of explainability, and misinterpretation of null values. The sparsity is the phenomenon that users, in general, interact with only a few numbers of items. Lack of explainability in recommender systems is the inability of justifying the predicted list of items to a user. Finally, the misinterpretation of null values can produce inaccurate predictions since the null values can represent an unseen or a disliked item for the user.

To address the research problems described above, we exploit explicit and implicit data from online systems in order to predict a list of items to a user to suggest items according to the user's personal preferences. We propose several methods and algorithms to learn the user's preferences and predict the most relevant items for the user via multi-view latent factor models. Additionally, we propose the explainable method to interpret the recommended list of items.

To evaluate the proposed methods, we performed an experimental evaluation of nine benchmark datasets: LDOS-CoMoDa, InCarMusic, Frappe, Movielens, Foursquare, Wee, Amazon, Yelp, and Pinterest. The results of the detailed data analysis and empirical experiments present the high accuracy and effectiveness in online systems of the proposed methods.

Resumé

Formålet med anbefalelsessystemer er, at forudsige de foretrukne varer for en bruger i henhold til hans/hendes præferencer. Den mest populære teknik er kollaborativ filtrering, der er klassificeret som hukommelses- eller modelbaseret. Den modelbaserede teknik sigter mod, at identificere skjulte funktioner gennem maskinindlæringsmetoder. Blandt de modelbaserede tilgange er latente faktor modeller populære på grund af deres høje nøjagtighed i, at forudsige brugerens præferencer. Imidlertid har de problemer med sparsomme data, forklaringsvanskeligheder og fejlfortolkede nulværdier. Årsagen til sparsomme data er, at brugerne kun har interageret med få varer. Forklaringsvanskeligheder skyldes, at den forudsagte liste af varer ikke kan retfærdiggøres overfor brugeren. Endelig kan fejlfortolkningen af nulværdier frembringe unøjagtige forudsigelser, da nulværdierne kan repræsentere en ukendt eller disliked genstand for brugeren.

For at løse de ovenfor beskrevne forskningsproblemer anvender vi de eksplicitte og implicitte data fra online-systemer til, at forudsige en liste med varer til brugeren, i henhold til brugerens personlige præferencer. Vi foreslår flere algoritmer og metoder til at lære brugerens præferencer at kende og forudsige de mest relevante varer for brugeren gennem multi-view latente faktormodeller. Desuden foreslår vi en forklaringsmetode til at fortolke den anbefalede liste af varer.

Til at evaluere de foreslåede metoder, udførte vi en eksperimentel evaluering af ni benchmark datasæt: LDOS-CoMoDa, InCarMusic, Frappe, Movielens, Foursquare, Wee, Amazon, Yelp og Pinterest. Resultatet af de detaljerede dataanalyser og empiriske eksperimenter viser en større nøjagtighed i online anbefalinger og dermed effektiviteten af de foreslåede metoder.

Acknowledgments

This thesis is dedicated to the memory of my father, Francisco da Costa. I miss him, but I am glad to know he is happy in a better place.

Thanks for the help and support of my friends and colleagues. Most of all, I would like to express my sincere gratitude to my Ph.D. supervisor, Professor Peter Dolog. He has always been supportive and committed throughout my Ph.D. journey. Thank you for your advice, concrete and constructive contributions.

I would like to thank all my colleagues at the Computer Science department at Aalborg University for all discussions, insights, and support during this journey. Special thanks to all colleagues who played table football with me. The games helped me to get new energy whenever needed. Thanks to secretaries and technicians of the Department for the incredible and prompt support.

I wish to extend my thanks to all the colleagues at the Insight Centre for Data Analytics in Dublin. Special thanks to Professor Aonghus Lawlor and Ph.D. student Sixun Ouyang for the discussions and insights during my stay as a visiting student.

To my family and friends, I thank you all for making the experience of being abroad memorable. Special thanks to my former flatmates for your friendship. I thank in particular Nanna for the patience, support, and energy to raise my mood when I most needed it. Thanks to Nanna's family for the constant help. My mother, Maristela, for the unconditional love. Furthermore, Marianne, Aquiles, Aluey, and Laercio for bringing joy.

Felipe Soares da Costa,
Aalborg, Denmark, 2019

Contents

Thesis Details

Thesis Title:	Multi-view Latent Factor Models for Recommender Systems
Ph.D. Student:	Felipe Soares da Costa
Supervisor:	Assoc. Prof. Peter Dolog, Aalborg University

The main body of this thesis consists of the following accepted and submitted papers.

[A] F. Costa and P. Dolog, "Hybrid learning model with barzilai-borwein optimization for context-aware recommendations". *In Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida USA, pages 456–461, 2018.*

[B] Muhammad Aamir Saleem, Felipe Soares da Costa, Peter Dolog, Panagiotis Karras, Toon Calders and Torben Bach Pedersen. "Predicting Visitors Using Location-Based Social Networks". *In Proceedings of 19th IEEE International Conference on Mobile Data Management, MDM'18, Aalborg, Denmark, pages 245–250, 2018.*

[C] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations". *In Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion, IUI 2018, Tokyo, Japan, pages 57:1–57:2, 2018.*

[D] Felipe Costa and Peter Dolog, "Neural Explainable Collective Non-negative Matrix Factorization for Recommender Systems". *In Proceedings of the 14th International Conference on Web Information Systems and Technologies, WEBIST 2018, Seville, Spain, pages 35–45, 2018.*

[E] Felipe Costa and Peter Dolog, "Collective Embedding for Neural Context-Aware Recommender Systems". *Under revision for a Conference publication, submitted in January, 2019.*

[F] Felipe Costa and Peter Dolog, "Convolutional Adversarial Latent Factor Model for Recommender System". *To be published in Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, FLAIRS 2019, Sarasota, Florida USA, 2019.*

This thesis has been submitted for assessment in partial fulfillment of the Ph.D. degree. The thesis is based on the submitted and pusblished scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extend summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis has been obtained from the corresponding publishers with the conditions that they are cited and the DOI pointers and /or copyright/credits are placed prominently in the references.

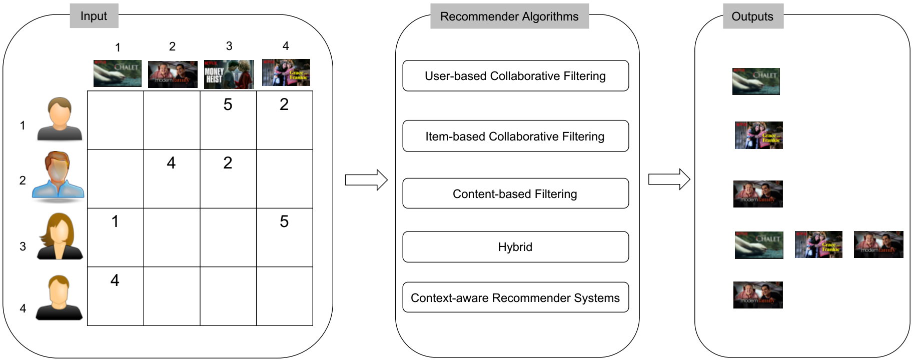
Part I

Thesis Summary

Introduction

1 Background and Motivation

Information overload is a phenomenon experienced by online users nowadays. To overcome this problem, Recommender Systems (RS) aim to provide a list of preferred items to a user based on her/his preferences. A broad range of online services such as e-commerce, e-learning, music and movie streamings, and tourism among others use RS in their systems. Figure 1.1 illustrates the general framework of RS.



	1	2	3	4
1			5	2
2		4	2	
3	1			5
4	4			

Input
Recommender Algorithms
User-based Collaborative Filtering
Item-based Collaborative Filtering
Content-based Filtering
Hybrid
Context-aware Recommender Systems
Outputs

Fig. 1.1: General Framework of Recommender Systems

The left side in Figure 1.1 has the input elements: users, items and ratings. The middle contains RS algorithms, which generates different predictions depending on the chosen method. The right side presents the outputs of each algorithm with a list of the recommended items. As an example, the first user 1 prefers action TV series. Considering the user-based collaborative filtering algorithm, the only user who prefers action series is the user 4. Based on the similarity of both users, the algorithm recommends the TV series watched by the user 1 to user 2.

The most well-known technique from RS is Collaborative Filtering (CF), which recommend items based on users or items similarities. Researchers mainly classify the CF technique as memory- and model-based. The memory-based approach relies on similarity measures such as cosine similarity, Pearson correlation, or Jaccard coefficient to identify similar users or items. On the other hand, the model-based approach relies on identifying hidden features through machine learning methods, such as Matrix Factorization (MF).

MF is a Latent Factor Model (LFM) which provides high accuracy in the RS research area. Primarily, MF considers the rating prediction task based on explicit data, where the reasoning relies on previous user-item interactions, such as ratings. However, explicit data is hard to retrieve, due to low user engagement in the online system. Recently, MF focus on implicit data aiming to recommend a sorted list of N items rather than rating predictions due to the facility in retrieving the data from online systems.

A research problem faced by RS is data sparsity, which denotes a matrix with most of the values equal to zero. The sparsity problem in RS happens due to the user-item interactions for observed items in comparison with unobserved items is exponentially low for explicit and implicit data. To solve the data sparsity problem, researchers have proposed to incorporate contextual information in some applications during the recommendation process to increase the accuracy of the recommendation models [1]. Examples of context in different fields are sights to be visited (tourism), time and place (movies), and so on. The quality of recommendations increases when online systems utilize additional information, such as *time* and *location* [1].

Furthermore, other research problems have emerged recently such as explainability of recommendation models and misinterpretation of null values on implicit data. The explainability of recommendations has become an important task due to a lack of interpretability for the recommended top-N list of items, which may decrease the users' fidelity to a system. The null values mislead the RS to misinterpret the recommendations as they can mean that user dislikes or have not previously seen the item. The misinterpretation may cause lower accuracy and effectiveness of the recommender system.

2 Research Problems

The research problems covered by this thesis are described as follows:

2.1 Sparsity in Recommender Systems

The user-item interactions are modeled considering the observed interactions, such as ratings or purchasing as described in Section 1. For example, in an online movie streaming service with a million movies in their catalog, each user is represented by a feature vector of a million entries. The value for each entry is defined by whether the user has watched the target movie previously. For explicit data, we observe ratings from 1 to 5 indicating user interactions or 0 when there are no interactions. Similarly, for implicit data, it is denoted 1 when a user interacts with an item and 0 for unknown interactions. A user-item interaction matrix is built based on those interactions from multiple users and items as illustrated in Figure 1.2.

items
users
items
users

3	0	0	4
0	0	0	5
0	0	0	3
0	1	0	2

(a) Explicit data

1	0	0	1
0	0	0	1
0	0	0	1
0	1	0	1

(b) Implicit data

Fig. 1.2: User-item interaction matrix

The amount of unobserved items is usually higher than the observed items due to the online platforms present a vast amount of content in their catalog. The task of the user interacting with all the items in the online catalog is complicated, resulting in a large amount of non-interacted items computationally represented by 0. This problem is known as the sparsity problem and presents a negative impact on the accuracy of the collaborative filtering method, due to the lower probability of identifying similarities among users or items. Moreover, it causes slow time convergence during the learning step to predict users' preferences.

2.2 Explainability in Recommender Systems

The current RS methods sufficiently disclose nor explain the reason for the predicted list of items given to a target user, generating problems such as lack of trust and transparency in the online services. Recently, Zhang *et al.* [37] formally introduced the term *explainable recommendations*. The goal of explainable recommendations is to generate interpretable models capable of explaining the recommendation results humanly. The literature presents different strategies to solve the explainability research problem such as visual- or graph-based explanations, however, in this thesis, we focus on the text-based strategy.

The explanations for recommendations may have a substantial contribution to the acceptance and success of the RS from the user's perspective. The goal is to improve six different aspects:

1. Effectiveness: explanations may support user decisions by helping them to understand why they like or dislike a particular recommended item.

2. Efficiency: explanations may support a quick user decision, due to an objective and structured way of interpreting the recommendations.

3. Persuasiveness: explanations may help the online system to persuade a user to interact with a target item.

4. Satisfaction: explanations may increase the user's satisfaction with a given list of items.

5. Transparency: explanations clarify how a recommendation was chosen, making the online system transparent to the user.

6. Trust: explanations may increase the trust and fidelity of a user.

The online platforms contain users' reviews as a detailed source of information about the users' preferences. The research problem focuses on using review-based text to explain recommendations considering the six aspects mentioned above.

2.3 Null Values in Recommender Systems

Considering the implicit data scenario and the statement that users tend to interact with a few of the items available in the online system's catalog, the interpretation of unobserved items becomes hard. Unseen items denote that: (1) the user does not know the item exists, or (2) dislike the item.

The results provided by the RS based on the user's previous behavior may decrease its accuracy and efficiency if it models the null values as disliked items. Researchers have made efforts to improve the recommendations considering the null values. However, it is yet a challenge, especially in the implicit data scenario.

3 Evaluation Methods

The goal of the proposed methods in this thesis is to improve the prediction for top-N recommendations. To measure the quality of the recommendations, we use different metrics: precision, recall, f-measure, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [3]. We calculate the precision according to Equation 1.1 to measure how relevant the recommended list of items are for the user [21].

$$\text{precision} = \frac{TP}{TP + FP}, \tag{1.1}$$

where TP represents all recommended interacted items and FP denotes all recommended non-interacted items.

We compute recall to measure the probability of recommending the appropriate item as shown in Equation 1.2 [21] .

$$\text{recall} = \frac{TP}{TP + FN}, \tag{1.2}$$

where TP has the same semantics as in Equation 1.1 and FN denotes all non-interacted items that were not recommended.

We measure the harmonic mean between precision and recall via f-measure as defined in Equation 1.3 [3].

$$\text{f-measure} = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.3}$$

To compute the robustness of our models considering the ranked list of items, we applied HR as shown in Equation 1.4 [20].

$$Hit@N = \frac{NumberofHits@N}{|GT|}, \tag{1.4}$$

where GT denotes the ground-truth test set. We denote a *hit* when an item from the test set appears in the recommended list of items.

To measure the quality of the ranked list by considering the position of the item in the list via NDCG as defined in Equation 1.5 [20].

$$NDCG@N = Z_N \sum_{p=1}^{N} \frac{2^{r_p} - 1}{log_2(p+1)}, \tag{1.5}$$

where Z_N denotes the normalized values between 0 and 1; and r_p denotes relevance of item at position **p**. We define $r_p = 1$ if the item is in the test set, and 0 otherwise.

The empirical evaluation follows different strategies for each method described in Paper A, Paper B, Paper C, Paper D, Paper E, and Paper F. A brief explanation of each paper is given in Section 4.

We evaluated the quality of recommendations in Paper A, and Paper D using the k-fold cross-validation method with k equals to 5.

To evaluate the quality of the predictions given by the method proposed in Paper B, we split the dataset per month, where the first eight months correspond to the training set, and the remaining are the test set.

We applied an adapted version of the leave-one-out method proposed by Koren *et al.* [26] to evaluate the methods described in Paper E and Paper F. We held-out the latest user-item interaction as the test set and sample the remaining randomly 100 items which were not interacted by the users as the training set.

To evaluate the quality of the explanations in Paper C, we applied the readability metrics presented by Maharjan *et al.* [28]: Automated Readability

Index (ARI) [34], Flesch-Kincaid Grade Level (FGL) [25], Gunning Fog Index (GFI) [16], Simple Measure of Gobbledygook (SMO) [29], Coleman-Liau Index (CLI) [7], Lesbarhets Index (LIX) [2], and Rate Index (RIX) [2]. Table 1.1 summarizes the definition and computation of each metric.

Readability Metric	Description and Equation
ARI	Measures the understandability of a text. The output represents the U.S grade level necessary to comprehend a text. $ARI = 4.71 \times (\frac{\text{characters}}{\text{words}}) + 0.5 \times (\frac{\text{words}}{\text{sentence}}) - 21.43$
FGL	Measures how difficult a sentence is to understand. The output number corresponds with a U.S grade level. $FGL = 0.39 \times (\frac{\text{total words}}{\text{total sentences}}) + 11.8 \times (\frac{\text{total syllables}}{\text{total words}})$
GFI	Measures the readability of a of English writing sample. The output indicates the number of years of formal education (U.S grade) that a person requires to understand the text on the first reading. $GFI = 0.4 \times (\frac{\text{words}}{\text{sentence}}) + 100 \times (\frac{\text{complex words}}{\text{words}})$
SMO	Measures the understandability of a text. The output represents the number of years of U.S. grade level to comprehend the text. $SMO = \sqrt{\text{total complex words} \times (\frac{30}{\text{total sentences}})} + 3$
CLI	Computes the understandability of a text. The output represents the U.S grade level necessary to comprehend a text. $CLI = (5.89 \times (\frac{\text{characters}}{\text{words}})) - (30 \times (\frac{\text{sentences}}{\text{words}})) - 15.8$
LIX	Computes the readability of a text. The output represents a grade level, where index below 0.1 denotes grade 1 and above 7.2 denotes college. $LIX = (\frac{\text{total words}}{\text{total sentences}}) + (\frac{\text{long words}}{\text{total words}} \times 100)$ where long words are words with more than six characters.
RIX	Computes the readability of a text. The output represents how difficult the text is, where score is between 0 (very easy) and 55 (very difficult). $RIX = (\frac{\text{long words}}{\text{sentences}})$ where long words are words with more than six characters.

Table 1.1: Readability Metrics

Additionally, Paper C and Paper D presents examples of the explanations generated by the proposed method as part of the evaluation.

4 Organization

The thesis is organized according to the structure given by Figure 1.3. Chapter 2 summarizes Paper A and describes a new model named CHNMF to obtain top-*N* recommendations based on users-items interactions, items, and contextual features. Chapter 3 summarizes Paper B and describes a new model named CMViP to predict visitors with a higher probability for visiting a target place. Chapter 4 summarizes Paper C and describes a review-oriented text generation to explain recommendations. Chapter 5 summarizes Paper D and introduces a new model named NECoNMF, which explores explanation-based recommendations through LFM by extending Paper A, and Paper C. Chapter 6 summarizes Paper E and describes a new model named CoNCARS, exploring the hybridization of the neural network and MF using time as important information for recommendations. Chapter 7 summarizes Paper F and describes a new model named CALF, exploiting the adversarial training to improve top-*N* recommendations. Chapter 8 gives a summary of the contributions of the papers in this thesis. Finally, Chapter 9 concludes and proposes future directions for the Ph.D. thesis.

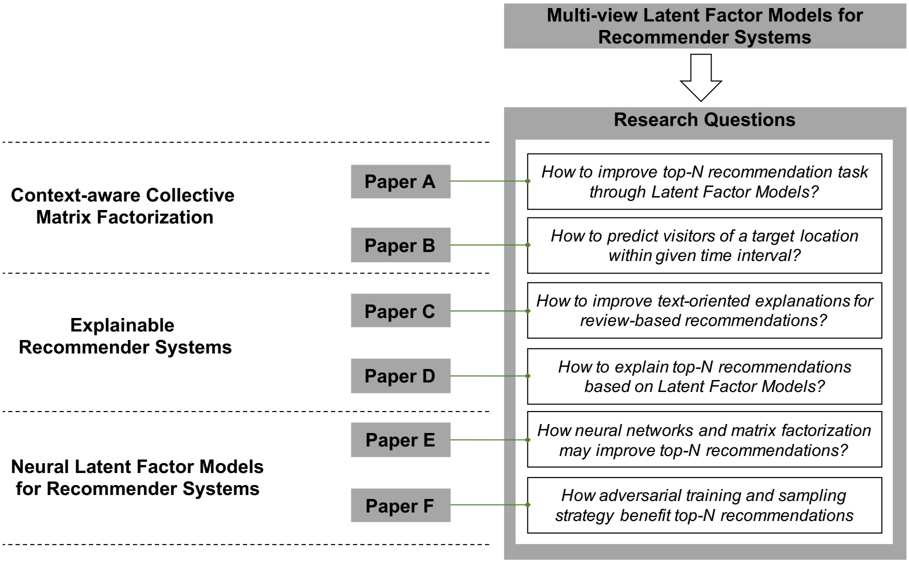

Multi-view Latent Factor Models for Recommender Systems
Research Questions
Context-aware Collective Matrix Factorization
Paper A
Paper B
How to improve top-N recommendation task through Latent Factor Models?
How to predict visitors of a target location within given time interval?
Explainable Recommender Systems
Paper C
Paper D
How to improve text-oriented explanations for review-based recommendations?
How to explain top-N recommendations based on Latent Factor Models?
Neural Latent Factor Models for Recommender Systems
Paper E
Paper F
How neural networks and matrix factorization may improve top-N recommendations?
How adversarial training and sampling strategy benefit top-N recommendations

Fig. 1.3: Thesis structure.

Collective Matrix Factorization for Top-N Recommendation

This Chapter gives an overview of Paper A [8].

1 Motivation and Problem Statement

LFM has improved modern RS through accurate methods. Non-negative Matrix Factorization (NMF) is an LFM extensively applied in RS, which utilize non-negative values during the matrix decomposition. Nonetheless, NMF does not consider multi-view recommendations, becoming inefficient for Context-aware Recommender Systems (CARS). To solve this issue, researchers have been investigating Collective Matrix Factorization (CMF) methods. CMF aims to collective factorize multiple user-item feature interactions to improve the top-N recommendations and minimize the sparsity problem. However, CMF has neglected two issues: contextual features and slow learning convergence. We propose the Collective Hybrid Non-negative Matrix Factorization model (CHNMF) [8] to optimize the recommendation algorithm. The model collectively factorizes explicit user-item interactions, contextual information, and content features into three non-negative low-rank matrices and common latent space. Jointly factorizing multi-view features maximize the accuracy for top-N recommendation and solve the sparsity problem.

Moreover, CHNMF extends the CMF utilizing the Barzilai-Borwein optimization method and Multiplicative Update Rules (MUR), improving the embedding learning convergence time. The hybrid model performs better in dense matrices. Hence Barzilai-Borwein optimization computes two projections and two gradients at even steps and determines the step length without using any line search [8]. The Paper A addresses the challenges above applying: 1) an efficient hybrid embedding learning method, combining Barzilai-Borwein optimization and MUR; 2) a collective learning feature method considering contextual features, content features, and user-item explicit interactions. We formally define the problem statement as [8] : *Given a user* **u***, an item* **i***, a content* **a***, and a context* **c** *predict a list of items with size N for user* **u**.

2 Collective Hybrid Non-negative Matrix Factorization Model

To provide the CHNMF model, we exploit the entities of general online systems: users-items interactions, content features, and contextual information. Based on these features, we segregate the data into three matrices, the user-item interaction matrix ($\mathbf{X}_u$), the content features ($\mathbf{X}_a$), and the contextual information ($\mathbf{X}_c$). These matrices are defined below reproduced from [8].

Definition 2.1. *Interaction matrix ($\mathbf{X}_u$) represents the item's ratings given by a user* **u***. The user-item matrix is modelled as $\mathbf{X}_u = \{x_u \in \mathbb{R}^{u \times i} | 1 \leq x_u \leq 5\}$, where* **u** *is the number of users,* **i** *is the number of items.*

Definition 2.2. *Content feature matrix ($\mathbf{X}_a$) represents whether a user* **u** *interacted with an item's content feature* **a***. The user-content feature matrix is formally defined as $\mathbf{X}_a = \{x_a \in \mathbb{Z}^{u \times a} | 0 \leq x_a \leq 1\}$, where* **u** *is the number of users and* **a** *is the content size.*

Definition 2.3. *Contextual information matrix ($\mathbf{X}_c$) represents how often a user* **u** *interacted with an item's content feature* **c***. The user-contextual information matrix is is formally defined as $\mathbf{X}_c = \{x_c \in \mathbb{Z}^{u \times c} | 0 \leq x_c \leq 1\}$, where* **u** *is the number of users and* **c** *is the context a user rated an item.*

The context frequency measures how often the users rated the items in specific circumstances. Some users may have higher absolute frequency than others due to rate a larger set of items. Therefore, we need to normalize the context frequency. Given a set of items I and a set of contextual information C, where $\mathbf{i}_c$ is an item rated in a specific context, we can formalize the frequency as [8]:

$$CF_i = \frac{\sum\limits_{\mathbf{i}=1}^{n} \mathbf{i}_c}{\sum\limits_{\mathbf{i}=1}^{n} \mathbf{i}}, \tag{2.1}$$

Next, CHNMF decomposes the input matrices into two low-rank approximation matrices. The user-item interaction matrix is decomposed as $\mathbf{W} \times \mathbf{H}_u$; categories as $\mathbf{W} \times \mathbf{H}_a$; and context $\mathbf{W} \times \mathbf{H}_c$. Where, $\mathbf{H}_u$ denotes the latent features for user **u**. Similarly, $\mathbf{H}_a$ represents the category's latent features **a**, and $\mathbf{H}_c$ represents the context's latent features **c**. Finally, **W** represents the common latent space. The goal is to solve the following optimization problem [8]:

$$\begin{aligned} min: f(\mathbf{W}) = \frac{1}{2}[\alpha\|\mathbf{X}_u - \mathbf{W}\mathbf{H}_u\|_2^2 + \beta\|\mathbf{X}_a - \mathbf{W}\mathbf{H}_a\|_2^2 \\ +\gamma\|\mathbf{X}_c - \mathbf{W}\mathbf{H}_c\|_2^2 \\ +\lambda(\|\mathbf{W}\|_2 + \|\mathbf{H}_u\|_2 + \|\mathbf{H}_a\|_2 + \|\mathbf{H}_c\|_2)] \\ s.t.\mathbf{W} \geq 0, \mathbf{H}_u \geq 0, \mathbf{H}_a \geq 0, \mathbf{H}_c \geq 0 \end{aligned} \tag{2.2}$$

where $\{\alpha, \beta, \gamma\} \in [0, 1]$ are hyper-parameters controlling the importance of each factorization and the hyper-parameter $\lambda \geq 0$ is used to enforce a smooth solution and avoid overfitting.

3 Hybrid Learning Model

CHNMF is classified as a non-convex problem when considering all the decomposed matrices together ($\mathbf{W}$, $\mathbf{H}_u$, $\mathbf{H}_a$, $\mathbf{H}_c$) [8]. Solutions have been proposed in the literature to solve this problem, such as MUR proposed by Saveski *et al.* [33]. However, Huang *et al.* [22] demonstrates that MUR converges relatively slowly to find the optimal solution. As an alternative, CHNMF utilizes a hybrid embedding learning model using Barzilai-Borwein optimization and MUR method to solve the time convergence challenge. Barzilai-Borwein optimizes the matrices $\mathbf{H}_u$, $\mathbf{H}_a$, and $\mathbf{H}_c$ through the algorithm, while MUR updates the matrix $\mathbf{W}$ as described in Paper A.

Algorithms	**LDOS-CoMoDa**			**InCarMusic**			**Frappe**			**Movielens**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
UISplitting	0.0006	0.0030	0.0010	0.0035	0.0175	0.0058	0.0094	0.1560	0.0177	NA[1]	NA[1]	NA[1]
DSPF	0.0138	0.0690	0.0230	0.1008	0.0504	0.0672	0.00261	0.0984	0.0412	NA[1]	NA[1]	NA[1]
CAMF-C	0.0008	0.0043	0.0013	0.0045	0.0229	0.0075	0.1384	0.5582	0.2218	0.0003	0.0008	0.0004
CSLIM-ICS	0.0026	0.0131	0.0043	0.0031	0.0159	0.0051	NA[1]	NA[1]	NA[1]	0.0943	0.0257	0.0403
CSLIM-LCS	0.0031	0.0157	0.0051	0.0049	0.0247	0.0081	NA[1]	NA[1]	NA[1]	0.0018	0.0005	0.0009
CLSIM-MCS	0.0023	0.0117	0.0038	0.0028	0.0143	0.0046	NA[1]	NA[1]	NA[1]	0.0017	0.0004	0.0006
LCE	0.1268	0.1368	0.1316	0.2111	0.1874	0.1985	0.5952	0.5749	0.5848	0.2000	0.1900	0.1948
CoNMF	0.1254	0.1467	0.1352	0.1943	0.1563	0.1732	0.5888	0.5735	0.5810	0.1988	0.1805	0.1892
MultiNMF	0.1305	0.1775	0.1504	0.1867	0.1743	0.1803	0.5830	0.5731	0.5780	0.1901	0.1800	0.1849
CHNMF	**0.1373**	**0.2033**	**0.1639**	**0.2222**	**0.1996**	**0.2103**	**0.5986**	**0.5763**	**0.5872**	**0.2032**	**0.1989**	**0.2010**

Table 2.1: Top-5 Recommendations [8]

The Barzilai-Borwein optimization method aims to solve the following problem as [8]:

$$\min_{\mathbf{W} \geq 0} : f(\mathbf{W}, \mathbf{H}) = \frac{1}{2}\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 \tag{2.3}$$

The function $P(.)$ maps the negative values into zero. Knowing that $\mathbf{H}$ is a stationary point of Equation 2.3 for any $\alpha > 0$ as described by Costa *et al.* [8], then,

Algorithms	LDOS-CoMoDa			InCarMusic			Frappe			Movielens		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
UISplitting	0.0011	0.0111	0.0020	0.0062	0.0628	0.0112	0.0004	0.0032	0.0007	NA[1]	NA[1]	NA[1]
DSPF	0.0005	0.0055	0.0009	0.0070	0.0707	0.0127	0.0003	0.0023	0.0005	NA[1]	NA[1]	NA[1]
CAMF-C	0.0009	0.0094	0.0016	0.0073	0.0735	0.0132	0.0006	0.0046	0.0010	0.0017	0.0008	0.0010
CSLIM-ICS	0.0024	0.0094	0.0038	0.0038	0.0380	0.0069	NA[1]	NA[1]	NA[1]	0.0471	0.0257	0.0332
CSLIM-LCS	0.0025	0.0255	0.0045	0.0037	0.0373	0.0067	NA[1]	NA[1]	NA[1]	0.0017	0.0009	0.0017
CLSIM-MCS	0.0024	0.0240	0.0043	0.0028	0.0287	0.0051	NA[1]	NA[1]	NA[1]	0.0017	0.0009	0.0011
LCE	0.1389	0.1189	0.1281	0.1999	0.1190	0.1491	0.2997	0.1599	0.2085	0.2100	0.1974	0.2035
CoNMF	0.1188	0.1366	0.1270	0.1983	0.1189	0.1486	0.2992	0.1444	0.1947	0.2005	0.1901	0.1951
MultiNMF	0.1364	0.1183	0.1267	0.1970	0.1183	0.1478	0.2981	0.1451	0.1951	0.2009	0.1807	0.1902
CHNMF	**0.1399**	**0.1191**	**0.1286**	**0.2091**	**0.1194**	**0.1520**	**0.3020**	**0.1639**	**0.2124**	**0.2181**	**0.2000**	**0.2086**

Table 2.2: Top-10 Recommendations [8]

Algorithms	LDOS-CoMoDa	InCarMusic	Frappe	Movielens
UISplitting	0.0032	0.0295	0.0004	NA[1]
DSPF	0.0050	0.0428	0.0012	NA[1]
CAMF-C	0.0034	0.0232	0.5716	0.0008
CSLIM-ICS	0.0122	0.0181	NA[1]	0.0034
CSLIM-LCS	0.0122	0.0254	NA[1]	0.0107
CSLIM-MCS	0.0116	0.0134	NA[1]	0.0011
LCE	0.3232	0.1013	0.8019	0.1119
CoNMF	0.3201	0.0947	0.6179	0.1001
MultiNMF	0.3227	0.0962	0.6111	0.1099
CHNMF	**0.3366**	**0.1080**	**0.8048**	**0.1221**

Table 2.3: NDCG Performance [8]

$$\|P[\mathbf{H} - \alpha \nabla f(\mathbf{H})] - \mathbf{H}\|_F = 0. \tag{2.4}$$

The gradient $\nabla f(\mathbf{W})$, of $f(\mathbf{H})$, is Lipschitz continuous [14] with constant $L = \|\mathbf{W}^T\mathbf{W}\|_2$. Since $\mathbf{W}^T\mathbf{W}$ is a $k \times k$ and $k \ll min\{m, n\}$, the Lipschitz constant L is not expensive to obtain [8]. The Barzilai-Borwein algorithm calculates the best scores for the latent features until reaching the stationary point.

4 Discussion

The experimental evaluation measures the quality of recommendations in four publicly available benchmarks datasets: LDOS-CoMoDa, InCarMusic, Frappe, and Movielens. The results are observed in Tables 2.1, 2.2 and 2.3 considering four different metrics: precision, recall, f-measure and NDCG. First, we evaluate the effectiveness of CHNMF in recommending top-N items by retrieving the top 5 and 10 items. The Tables 2.1, 2.2 and 2.3 present CHNMF has an overall better performance when compared to other CMF methods, demonstrating that combining user-item interactions, content features, and contextual information play an important role in the quality of

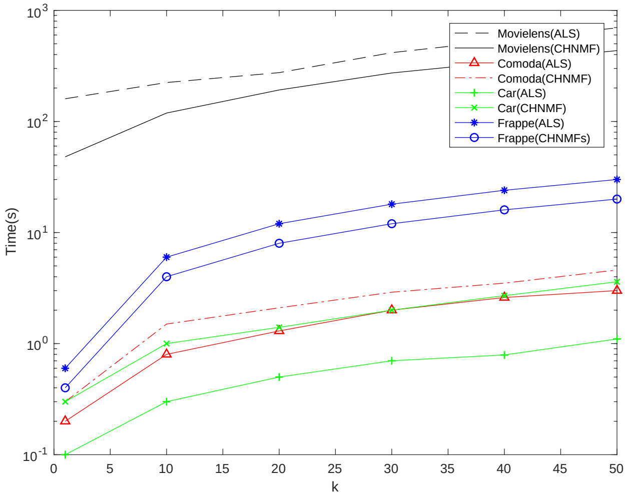


Fig. 2.1: Convergence Time [8]

recommendations. The CMF methods outperformed pre-filtering and contextual modeling techniques for CARS.

Moreover, CHNMF proved faster convergence time during the learning step in denser datasets such as Frappe and Movielens, when compared to Alternating Least Square (ALS) method as illustrated in Figure 2.1. The convergence time increases linearly according to the number of factors *k*.

Prediction of Visitors

This Chapter gives an overview of Paper B [32].

1 Motivation and Problem Statement

CHNMF improves the algorithm for RS, but it neglects the revenue expected by the system's owner. Considering that Location-Based Social Networks (LBSNs) provide data regarding user's preferences, such as check-ins and social relations, we aim to predict the users with higher probability to visit a target place, facilitating the management of applications, such as event planning. For example, a restaurant owner would like to forecast how many will attend the next planned event at her/his restaurant to organize better the event based on the expected amount of customers as illustrated in Fig 3.1. Paper B exploits the assumption of identifying potential visitors given a target location. We formally define the problem statement as [32]: *Given an LBSN, a location* **l***, and a time interval* **t***, predict the users who have a higher probability of visiting* **l** *within* **t**.

User	Location	Time
u_1	l_1	13:25 11/12/2017
u_1	l_1	14:30 12/12/2017
u_1	l_3	13:05 14/12/2017
u_2	l_1	13:10 11/12/2017
u_2	l_1	15:10 14/12/2017
u_3	l_1	14:20 11/12/2017
u_3	l_2	13:16 12/11/2017
u_3	l_3	16:20 14/12/2017
u_4	l_2	15:15 11/12/2017

Location	Category	Coordinates
l_1	Japanese	42.99,-71.46
l_2	French	42.98,-71.45
l_3	Japanese	42.97,-71.44

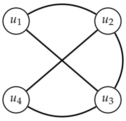


Fig. 3.1: Toy example adapted from [32] : Checkins (left), Location categories (top right) and social graph (bottom right)

2 Collective Matrix Factorization-based Visitor Prediction Model

The prediction of visitors requires to analyze LBSNs data by identifying which relevant features affect the mobility of users. Then, compute the probability of the users becoming potential visitors based on the CMViP prediction model. CMViP is a CMF which utilizes five relevant information that significantly contributes towards the prediction of user **u** visiting a location **l**. The features are: 1) visit frequency of **u** at **l** representing the number of check-ins a user **u** made in a location **l**; 2) the frequencies of visits of user **u** at locations with a similar category of **l**; 3) visit's frequencies of user **u** at locations at a given time interval **t**; 4) distance of **l** from the current location of **u**; and 5) influence of **u** on her/his followers.

The CMViP model follow three steps: applies CMF method to collectively factorizes the matrices based on four input features; then calculates the *visit score* ($\mathbf{Y}_S$) based on the linear model; and finally finds the potential visitors influenced by user **u**. The CMViP model computes the scores based on 1) visitor-location frequency matrix $\mathbf{Y}_U \in \mathbb{R}^{U \times L}$ denoting the entries as the number of visits of users U at locations L; 2) visitor-category matrix $\mathbf{Y}_C \in \mathbb{R}^{U \times C}$ denoting the entries as the number of visits of users for categories C; 3) visitor-time matrix $\mathbf{Y}_T \in \mathbb{R}^{U \times T}$ denoting the entries as the number of visits of users for each hour of the day; and 4) visitor-distance matrix $\mathbf{Y}_D \in \mathbb{R}^{U \times D}$ denoting the entries as the average distance users U travel for visiting locations L from their las locations. The matrices are decomposed into a common latent space matrix **W** and corresponding feature latent space matrix **H**. The formal definition of the optimization problem is defined by Equation 3.1 describe by Saleem et al. [32].

$$\begin{aligned} min : f(\mathbf{W}) = \frac{1}{2}[\alpha\|\mathbf{Y}_U - \mathbf{W}\mathbf{H}_U\|_2^2 + \beta\|\mathbf{Y}_C - \mathbf{W}\mathbf{H}_C\|_2^2 \\ +\gamma\|\mathbf{Y}_T - \mathbf{W}\mathbf{H}_T\|_2^2 + \eta\|\mathbf{Y}_D - \mathbf{W}\mathbf{H}_D\|_2^2 \\ +\lambda(\|\mathbf{W}\|_2 + \|\mathbf{H}_U\|_2 + \|\mathbf{H}_C\|_2 + \|\mathbf{H}_T\|_2 + \|\mathbf{H}_D\|_2)] \\ s.t. \mathbf{W} \geq 0, \mathbf{H}_U \geq 0, \mathbf{H}_C \geq 0, \mathbf{H}_T \geq 0, \mathbf{H}_D \geq 0 \end{aligned} \tag{3.1}$$

where $\{\alpha, \beta, \gamma, \eta\} \in [0, 1]$ are hyper-parameters controlling the importance degree of each matrix during the factorization. $\mathbf{H}_U$, $\mathbf{H}_C$, $\mathbf{H}_T$, and $\mathbf{H}_D$ are the latent feature matrices given by our input matrices. The hyper-parameter $\lambda \geq 0$ is applied to enforce the smoothness of the solution and avoid overfitting.

The CMViP model aims to find the users' visit scores given a location and a time interval. The linear equation combining the common latent space and the corresponding latent feature space matrices calculates the probability score for each user $\mathbf{u}$ visits location $\mathbf{l}$ considering all features. The *visit score* ($\mathbf{Y}_S$) is denoted by Equation 3.2 [32].

$$\mathbf{Y}_S(\mathbf{u},\mathbf{l},\mathbf{c},\mathbf{t},\mathbf{d}) = \alpha \times \mathbf{Y}_U(\mathbf{u},\mathbf{l}) + \beta \times \mathbf{Y}_C(\mathbf{u},\mathbf{c}) + \gamma \times \mathbf{Y}_T(\mathbf{u},\mathbf{t}) + \eta \times \mathbf{Y}_D(\mathbf{u},\mathbf{l}) \tag{3.2}$$

where α, β, γ, and η coefficients have the same meaning as in Equation 3.1 [32].

We provide a minimum visit score threshold θ. The users whose visit score is greater than θ are named as *Potential Visitors* $\mathbf{U}_P$ formally described by Equation 3.3 [32].

$$\mathbf{U}_P(\mathbf{l},\mathbf{c},\mathbf{t}) = \{\mathbf{u}|\mathbf{Y}_S(\mathbf{u},\mathbf{l},\mathbf{c},\mathbf{t},\mathbf{d}) \geq \theta\} \tag{3.3}$$

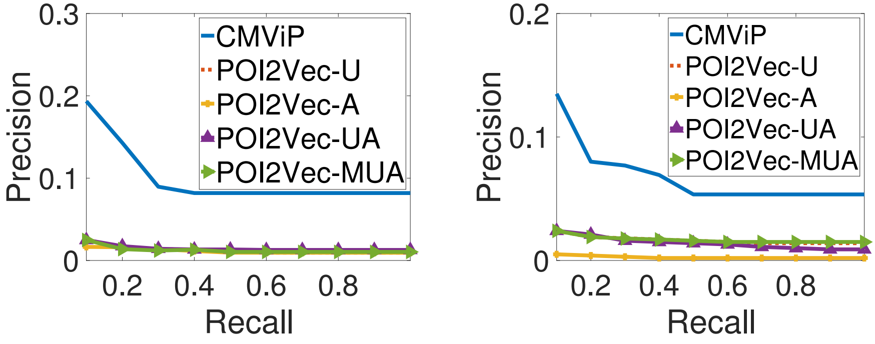
Based on the aforementioned model, we describe the following example:

> **Example 2.1**
> Consider the example given in Figure 3.1. Here, assume $t_1 = 13:00$, $\{\alpha,\beta,\gamma,\eta\} = 0.25$ giving equal importance degree to all features, and $\theta = 0.8$. Then, $\mathbf{Y}_S(u_1,l_1,c_1,t_1,d_1) = 0.84$. Similarly, $\mathbf{Y}_S(u_2,l_1,c_1,t_1,d_1) = 0.875$, $\mathbf{Y}_S(u_3,l_1,c_1,t_1,d_1) = 0.66$ and $\mathbf{Y}_S(u_4,l_1,c_1,t_1,d_1) = -\infty$. Thus, $\mathbf{U}_P(l_1,c_1,t_1,d_1) = u_1,u_2$.

Finally, CMViP incorporates another feature considering that users tend to follow their friends. Based on this statement, we assume that if a user visits a location, her/his friends may also join her/him. To figure out which friends have a higher probability to follow her/him, CMViP computes the influence of potential visitors on friends. The influenced potential visitors is named as *Influenced Potential Visitors* ($\mathbf{I}(\mathbf{U}_P)$) and calculated by using the Bernoulli distribution and a partial credit distribution based discrete time constrained model as formulated in Equation 3.4 [32].

$$\mathbf{I}(\mathbf{U}_P) = \{\mathbf{u}| \sum_{\mathbf{u}\in\mathbf{U}_P} \mathbf{Y}(\mathbf{u},\mathbf{v}) \geq \xi \wedge (\mathbf{u},\mathbf{v}) \in F\} \tag{3.4}$$

where $p(\mathbf{u},\mathbf{v})$ is the influence probability of $\mathbf{u}$ on $\mathbf{v}$, ξ is a threshold denoting the minimum probability of user influencing his/her followers, and F is the set of friends pairs in the LBSN. Hence, the set of predicted potential visitors is given by $\mathbf{U}_P \cup \mathbf{I}(\mathbf{U}_P)$, where $\mathbf{U}_P$ denotes the set of users having a significant visit score and $\mathbf{I}(\mathbf{U}_P)$ the potential visitors as illustrated by the following example:



(a) Wee

(b) Foursquare

Fig. 3.2: Precision-Recall Curve [32]

> **Example 2.2**
> Considering *Potential Visitors* given by the example 2.1 where $\mathbf{U}_P = \{u_1, u_2\}$. Morever, assume $\omega = 1h$ and $\eta = 0.2$. We observe for the location l_1, u_1 and u_2 are followed by u_3 and for the location l_3, u_1 is followed by u_3 within the given time window ω. Hence, the influence probability score of $\mathbf{U}_P$ on user u_3 is $p(\mathbf{U}_p, u_3) = 0.3 \geq \eta$ [32]. Finally, the set of the predicted visitors is u_1, u_2, u_3.

3 Discussion

The experimental setup to measure the performance of CMViP considers two real-world datasets: Foursquare and Wee. Foursquare is a smaller and denser dataset, while Wee is a larger and sparser dataset. The accuracy is computed based on the precision-recall curve to predict the potential visitors. To do so, we sort the dataset in ascending order of visiting time and divide it into two parts, the training set, and the test set. The training set is processed utilizing the CMViP model and further compared its results with the test dataset. The results compare CMViP with four variants of a state-of-art approach POI2Vec [15]. Analyzing the results shown in Figure 3.2 we observe CMViP outperforms POI2Vec up to 6 times.

Review-based Explanations for Recommender Systems

This Chapter gives an overview of Paper C [9].

1 Motivation and Problem Statement

The quality of predictions by RS has improved in the past years. However, other research problems have emerged such as explainability in recommendations. The explainable models aim to justify the reason behind the recommended list of items, making the online system more trustful. There are different alternatives to interpret a RS, for example, images or text. Images provide visual information regarding the prediction, while the text has more detailed information due to capturing the contextual information. The online systems provide item's reviews as a source of information regarding the quality of an item according to the user's point of view. Besides presenting the user's satisfaction regarding an item, the reviews may influence other users' decision of whether to interact or not with an item. Based on this assumption, we propose a review-oriented explanation model for RS in [9]. The proposed model described in Paper C utilizes a character-level attention-enhanced Long Short-Term Memory (LSTM) method to generate personalized natural language explanations based on user-generated reviews. The explanations are encoded considering the contextual information from the user-item interactions, for example, the explicit ratings. We formally define the problem statement as [9]: *Given a review* $\mathbf{s}$ *to a given item* $\mathbf{i}$ *from a user* $\mathbf{u}$*, and the corresponding rating* $\mathbf{r}$*, generate text to explain the recommended item.*

2 Explainable Model

The explainable model utilizes neural networks to process actual reviews and generate text according to the user's preferences. The character-level explanation model has three components named: LSTM network, attention layer, and generator module. The general interpretation model architecture is illustrated in Figure 4.1. Initially, the *doc2vec* method learns the users and items embeddings. Then, we concatenate the ratings to the vectors.

Further, we encode the characters from the items' reviews using one-hot encoding. Later, the embeddings are concatenated with the outputs from the LSTM network, becoming the input for the attention layer. Finally, the generator module decodes sentences to explain the recommendations.

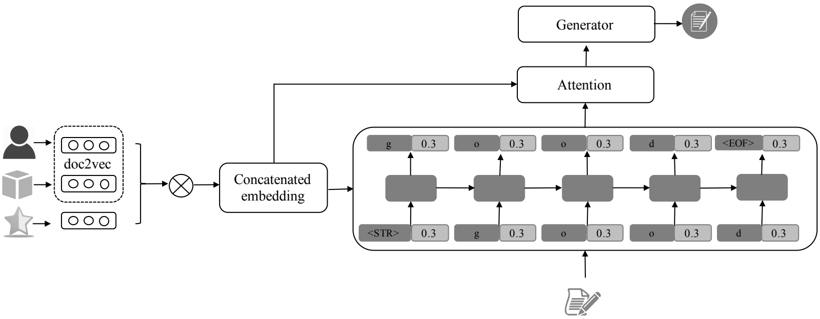


Fig. 4.1: Personalized Explanation Generation Model Architecture [9]

The LSTM network is one solution for sequence-based challenges, such as time and text generation. LSTM is an improvement to solve the vanishing gradient problem in Recurrent Neural Networks (RNN), which is a challenge for long-short term dependencies. The attention mechanism was adopted to adaptively learn soft alignments c_t between the character dependencies H_t and attention inputs a. c_t denotes the current cell state vector from the LSTM network. Equation 4.1 formally describes the new character dependencies given the attention layer $H_t^{attention}$ as described by Costa *et al.* [9].

$$\begin{aligned} c_t &= \sum_{i}^{a} \frac{\exp(\tanh(W_s \odot [H_t, a_i]))}{\sum \exp(\tanh(W_s \odot [H_t, a_i]))} a_i \\ H_t^{attention} &= \tanh(W_1 \odot c_t + W_2 \odot H_t) \end{aligned} \tag{4.1}$$

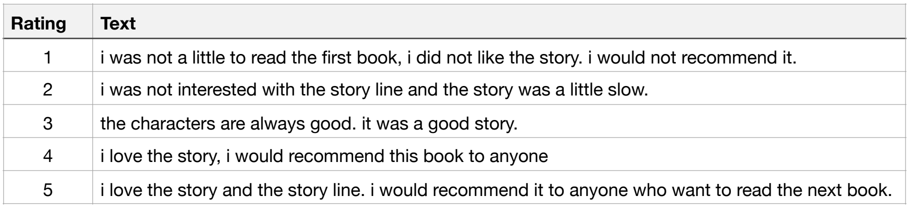
The generator module produces the explanation in character-level by maximizing the *softmax* probability p based on the new character dependencies as denoted by Equation 4.2 [9].

$$p = \mathrm{softmax}(H_t^{attention} \odot W + b), \quad char = \arg\max p \tag{4.2}$$

where **W** denotes the weight matrix and **b** the bias.

3 Discussion

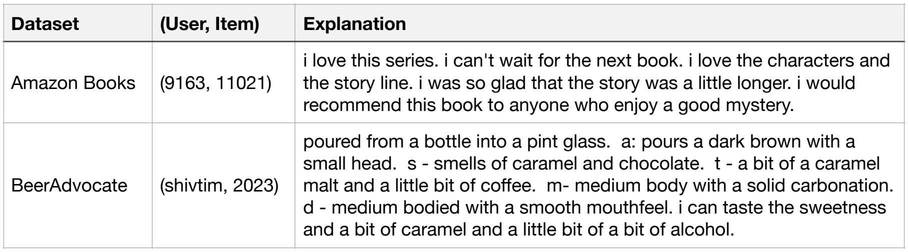
The experiments to measure the quality of explanations given by our model considers two real-world datasets: Beer-Advocate and Amazon book reviews. The first experiment measures the quality of generated explanations considering different ratings as attention mechanism as illustrated in Figure 4.2. Ratings close to 1 generate text with a negative sentiment regarding the item quality, while ratings close to 5 creates a text with positive sentiment.



Rating	Text
1	i was not a little to read the first book, i did not like the story. i would not recommend it.
2	i was not interested with the story line and the story was a little slow.
3	the characters are always good. it was a good story.
4	i love the story, i would recommend this book to anyone
5	i love the story and the story line. i would recommend it to anyone who want to read the next book.

Fig. 4.2: Rating Text Samples, from poorly rated (1) to highly rated (5) [9].

The second experiment measures the personalization task, where the explanation is generated given a user-item pair as presented in Figure 4.3. It is important to notice the generated text simulates a review which would be written by a target user about a given item.



Dataset	(User, Item)	Explanation
Amazon Books	(9163, 11021)	i love this series. i can't wait for the next book. i love the characters and the story line. i was so glad that the story was a little longer. i would recommend this book to anyone who enjoy a good mystery.
BeerAdvocate	(shivtim, 2023)	poured from a bottle into a pint glass. a: pours a dark brown with a small head. s - smells of caramel and chocolate. t - a bit of a caramel malt and a little bit of coffee. m- medium body with a solid carbonation. d - medium bodied with a smooth mouthfeel. i can taste the sweetness and a bit of caramel and a little bit of a bit of alcohol.

Fig. 4.3: Generated Text Samples [9]

Finally, the third experiment measures the quality of the generated text based on readability metrics described in Section 3. The goal is to measure how readable the generated text is. According to the metrics' description low score corresponds to easy and understandable text, while a high score

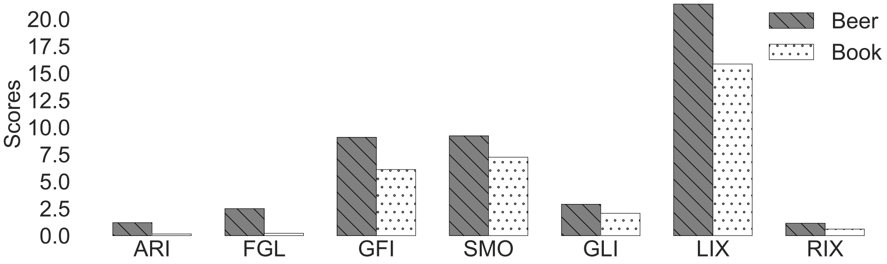


Fig. 4.4: Readability Score of Explainations [9]

denotes a more sophisticated writing style. Figure 4.4 illustrates the results considering the quality of the explanations generated by our model. Analyzing the Figure 4.4, we observe that, for the majority of the metrics, our method has stated the generated explanations are straightforward to read by students from early grades. The metric LIX, however, presents a high score, denoting the generated text is readable by college students. The result given by LIX is due to noise caused by the total amount of words considered in its equation.

Neural Explainable Latent Factor Model for Recommender Systems

This Chapter gives an overview of Paper D [10].

1 Motivation and Problem Statement

The LFMs described in Chapter 2 and 3 presented accurate results for RS. However, it is a challenge to explain *why* the online system predicts a particular list of items to a given user. One straight forward explanation follows the assumption: *The item* $\mathbf{i}$ *is recommended for user* $\mathbf{u}$*, due to its similarity with item* $\mathbf{j}$. However, items are defined by different aspects, for example, a movie has *horror* as its category, but due to its age classification, it is not recommended to watch with kids. The explanation has to be clear and cover the aspects the user like or dislike accordingly. The model proposed by Zhang *et al.* [37] named Explicit Factor Model (EFM) aims to use phrase-level sentiment analysis to explain the recommendation. The authors succeed in identifying the item's features which are more relevant to a target user, explaining the recommendation and disrecommendations. EFM uses a sentiment lexicon construction technique to extract the most relevant features from the reviews. Then, factorize the input information to identify the hidden features. However, EFM does not consider *how* to present the textual explanations while keeping their accuracy for top-N recommendation. Moreover, the EFM model does not add contextual information to the recommendation model. Considering this research challenge, we propose a Neural Explainable Collective Non-negative Matrix Factorization model (NECoNMF) to predict the user's preferences and explain the recommendation based on natural language generation [10]. The preferred features are identified applying: 1) CMF to jointly factorize four features: ratings, items' content features, contextual information and sentiments; and 2) an explainable module which explain the recommendation using LSTM to generate text based on the user's preferences [10]. We formally define the problem statement as [10]: *Given a user* **u**, *an item* **i**, *a content* **a**, *a context* **c**, *and a sentiment* **s** *predict a list of items with size N for user* **u** *and explain the recommendation.*

2 Neural Explainable Collective Non-negative Matrix Factorization Model

To predict the most relevant items given the user's preferences, we identify the features which affect the user's choices to recommend the top-N items. We propose the NeCoNMF model, which collectively factorize four different features: 1) ratings given to an item **i** from a user **u**; 2) interactions of user **u** with items having the same content features with **i**; 3) interactions of user **u** at a given contextual information **c**; and 4) the sentiment of a user **u** when she/he interacted with item **i**. The goal is to identify the probability of a user **u** liking or disliking an item **j** based on her/his previous feature interactions. Then, NECoNMF uses a neural generator model to explain the recommendations based on the features retrieved from the user's previous reviews aligned with the factorized features previously described.

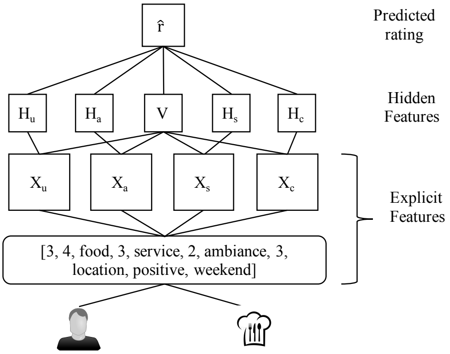


Fig. 5.1: Collective Non-Negative Matrix Factorization [10]

Consider the restaurant scenario as illustrated in Figure 5.1, where a user express her/his preference according to different aspects of a restaurant using numerical ratings and later writes a review with a detailed explanation. NECoNMF factorizes four different input matrices based on these user-item interactions: 1) content-feature frequency matrix $\mathbf{X}_a = \{x_a \in \mathbb{R}^{u \times a} | 0 \leq x_a \leq 1\}$ denoting the entries as the number of interactions of users for content **a**; 2) user-item interaction matrix $\mathbf{X}_u = \{x_u \in \mathbb{R}^{u \times i} | 1 \leq x_u \leq 5\}$ denoting the entries as the ratings the user **u** gave to item **i**; 3) user-context matrix $\mathbf{X}_c = \{x_c \in \mathbb{Z}^{u \times c} | 0 \leq x_c \leq 1\}$ denoting the entries as the number of interactions of users for contextual information **c**; and 4) and user-sentiment matrix $\mathbf{X}_s = \{x_s \in \mathbb{Z}^{u \times s} || x_s| = 1\}$ denoting the positive and negative sentiments from a user **u** to an item **i**. **u** is the number of users, **i** is the number of items, **a** is the content size, **c** is the context, and **s** is the sentiment regarding item features. The matrices are decomposed into a common latent space matrix **W** and corresponding feature latent space matrix **H**. The objective function to optimize is defined by Equation 5.1 [10].

$$\begin{aligned} min : f(\mathbf{W}) = \frac{1}{2}[\alpha \|\mathbf{X}_u - \mathbf{W}\mathbf{H}_u\|_2^2 + \beta \|\mathbf{X}_a - \mathbf{W}\mathbf{H}_a\|_2^2 \\ + \gamma \|\mathbf{X}_c - \mathbf{W}\mathbf{H}_c\|_2^2 + \omega \|\mathbf{X}_s - \mathbf{W}\mathbf{H}_s\|_2^2 \\ + \lambda(\|\mathbf{W}\|_2 + \|\mathbf{H}_u\|_2 + \|\mathbf{H}_a\|_2 + \|\mathbf{H}_c\|_2) + \|\mathbf{H}_s\|_2)] \\ s.t. \mathbf{W} \geq 0, \mathbf{H}_u \geq 0, \mathbf{H}_a \geq 0, \mathbf{H}_c \geq 0, \mathbf{H}_s \geq 0, \end{aligned} \tag{5.1}$$

where the hyper-parameters are defined by $\{\alpha, \beta, \gamma, \omega\} \in [0, 1]$ to control the importance degree of each matrix during the factorization. Defining the hyper-parameters as 0.25 gives equal importance to the matrices decomposition. The hyper-parameter $\lambda \geq 0$ is applied to control the smoothness of the solution and avoid overfitting.

3 Hybrid Learning Model

NeCoNMF extends the hybrid learning model described in Chapter 3 by updating the learning step for the sentiment matrix $\mathbf{H}_s$ using Barzilai-Borwein method.

4 Natural Language Explainable Model

The natural language explainable model aims to identify the most relevant features given by the user's reviews and further explain the predicted top-N recommended items. The model relies on the LSTM network to process and generate text. Further, to generate the explanation, the model has an attention layer to identify which features the user like or dislike. The detailed modules composing the explanation model are: 1) context encoder; 2) LSTM network; 3) attention layer; and 4) generator model [10].

The context encoder aims to translate the input character into one-hot encoding, and further concatenate the user's ratings given different item's content features. The formal definition is given in Equation 5.2 [10].

$$X_t' = [onehot(x_{char}); x_{aux}] \tag{5.2}$$

The output from the context encoder feeds the LSTM network that calculates the probability of which character comes after another. The goal is to build sentences on a character level, based on the encoded information. The attention layer is responsible for learning soft alignments h_t between character dependencies c_t and *attention* inputs, such as the ratings given by a user **u**. Equation 5.3 formally defines the character dependencies using attention layer $h_t^{attention}$ [10].

$$\begin{aligned} c_t &= \sum_{i}^{attention} \frac{\exp(\tanh(W_s \odot [h_t, attention_i]))}{\sum \exp(\tanh(W_s \odot [h_t, attention_i]))} attention_i \\ h_t^{attention} &= \tanh(W_1 \odot c_t + W_2 \odot h_t) \end{aligned} \tag{5.3}$$

Finally, the explanation is generated by the generator model, which concatenates an initial *prime* symbol to initiate the sentence according to Equation 5.4 [10].

$$p = \text{softmax}(H_t^{attention} \odot W + b), \quad char = \arg\max p \tag{5.4}$$

To formulate the sentence, we give the **prime** string **str**, then the Equation 5.4 calculates the probability of the next character until finding the predefined string **end** as proposed by Costa *et al.* [10]. The final string **end** controls the length of the generated text.

Dataset	**Yelp**				**Amazon**			
Metric	**NDCG@5**	**HIT@5**	**NDCG@10**	**HIT@10**	**NDCG@5**	**HIT@5**	**NDCG@10**	**HIT@10**
Top-N Algorithms								
ItemPop	0.0110	0.0136	0.0185	0.0306	0.0077	0.0082	0.0136	0.0238
PageRank	0.0235	0.0278	0.0313	0.0452	0.0978	0.1070	0.1029	0.1200
Explainable Recommendation								
TriRank	0.0258	0.0313	0.0353	0.0527	0.1033	0.1127	0.1086	0.1266
EFM	0.2840	0.0448	0.2955	0.0678	0.3615	0.1284	0.3670	0.1429
NECoNMF	**0.3366**	**0.0503**	**0.3461**	**0.0763**	**0.3892**	**0.1301**	**0.3962**	**0.1486**

Table 5.1: NDCG and Hit Ratio results for NECoNMF and compared methods at rank 5 and 10 [10]

5 Discussion

First, to measure the performance of NECoNMF we used two real-world datasets: Yelp and Amazon. The accuracy is computed based on the NDCG and HR metrics to predict the top-N items considering N equal to 5 and 10. The results were compared with four state-of-art approaches: ItemPop [13], PageRank [30], TriRank [17], and EFM [37]. Table 5.1 presents the results of the models, where we observe NECoNMF outperforms the state-of-art models in both datasets. NECoNMF improves the recommendations algorithms due to apply personalized historical information showing a better performance compared to the ItemPop method.

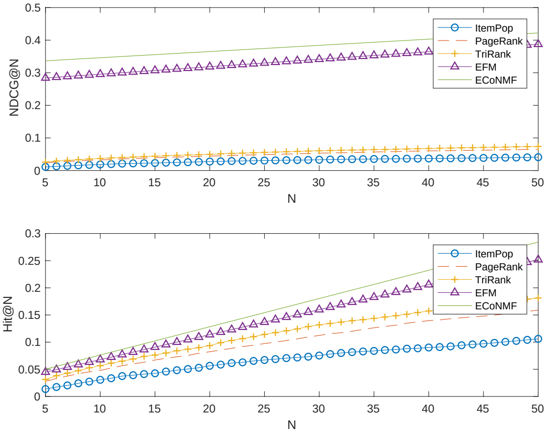


Fig. 5.2: Empirical evaluation on Yelp dataset for N from 5 to 50 [10]

Moreover, NECoNMF identifies hidden features based on the item's content features, context, sentiment, and rating improving the recommendation effectiveness when compared to PageRank and TriRank.

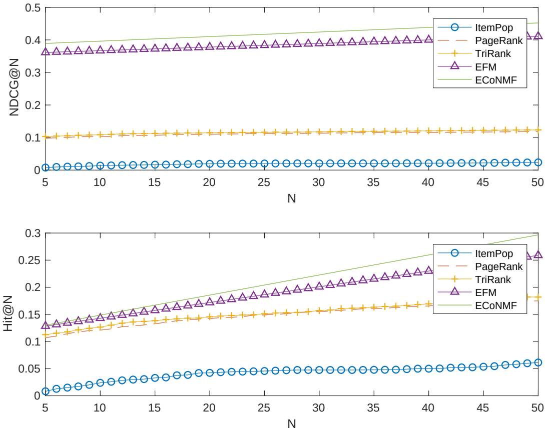


Fig. 5.3: Empirical evaluation on Amazon dataset for N from 5 to 50 [10]

Finally, NECoNMF factorizes two additional features: item's content features and contextual information when compared to EMF, allowing NECoNMF to better model the user's preferences in the vectorial space and improving

User	Item	Rating	Generated Explanation
A	X	1	i was disappointed. it was too small and noisy. food was good, but i just can’t come back with my family.
A	Y	3	this was a nice restaurant. i liked the service and the location. i’m not sure i’d recommend the food there.
B	X	5	i loved this restaurant. i could not leave it. i loved the service and the food. i will be back again next weekend.

Table 5.2: Example explanations produced by NECoNMF on the Yelp dataset [10].

the personalized recommendation task. Furthermore, NECoNMF collectively factorizes the input matrices into one common latent space among the matrices, allowing to identify hidden latent features not selected by the EFM model during the recommendation task.

Figure 5.2 and 5.3 illustrates an additional experiment to measure the NECoNMF’s performance for top-N recommendation task when N varies from 5 to 50 in comparison with the baseline models. NECoNMF outperforms the baselines in both datasets for NDCG and HR metrics. The performance increases linearly in all models, for example, a larger list of top-N items improves the recommendation’s accuracy.

An experiment was performed to analyze the quality of the generated explanations considering the natural language task as presented in Table 5.2. We conducted the experiment considering two users and two items with different predicted ratings. The generated texts are review-based explanations and present an overall proper interpretation of the recommended items. However, some improvements are necessary to offer better readability of the sentences.

Neural Latent Factor Model for Context-aware Recommender Systems

This Chapter gives an overview of Paper E [12].

1 Motivation and Problem Statement

The LFMs presented in Chapter 2, 3, and 5 describes the models which use context to overcome the sparsity problem, however the proposed models do not deeply exploit the *time* as feature. *time* is characterized as an important feature since users tend to either keep their preferences or change over time. Tensor Factorization (TF) has been presented as an important model to identify user preferences and provide accurate recommendations [23]. However, TF is not able to capture specific temporal patterns. Consider the example illustrated by Figure 6.1 where the user watches movies in different timestamps.

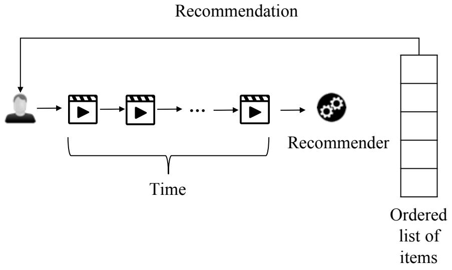


Fig. 6.1: Example for user behaviour over time. [12]

Initially, the user watches a sequence of movies in different timestamps. Then, the sequence of movies is given as input to the RS, which generates a list of recommended items. However, the RS model does not consider the changes in the user's preferences along the time. We propose the Collective Embedding for Neural Context-aware Recommender Systems (CoNCARS) model to solve the research problem defined by the previous example. CoNCARS presents the following layers: Input, Embedding, Pairwise Interaction, Hidden, Fusion, and Prediction. Section 2 describes each layer of the CoNCARS model. Paper E exploits the assumption of identifying whether user-item pair $(\mathbf{u}, \mathbf{i})$ should have a higher score than $(\mathbf{u}, \mathbf{j})$ for specific time $\mathbf{t}$. We formally define the problem statement as [12]: *Given a user* $\mathbf{u}$*, an item* $\mathbf{i}$*, and a timestamp* $\mathbf{t}$*, predict a list of items with size N for user* $\mathbf{u}$ *and verify whether the user* $\mathbf{u}$ *prefers item* $\mathbf{j}$ *over* $\mathbf{k}$.

2 Collective Embedding for Neural Context-Aware Recommender Systems

Considering the implicit data where the users prefer observed items over the unobserved items and time as a cyclic feature, we define CoNCARS as illustrated in Figure 6.2.

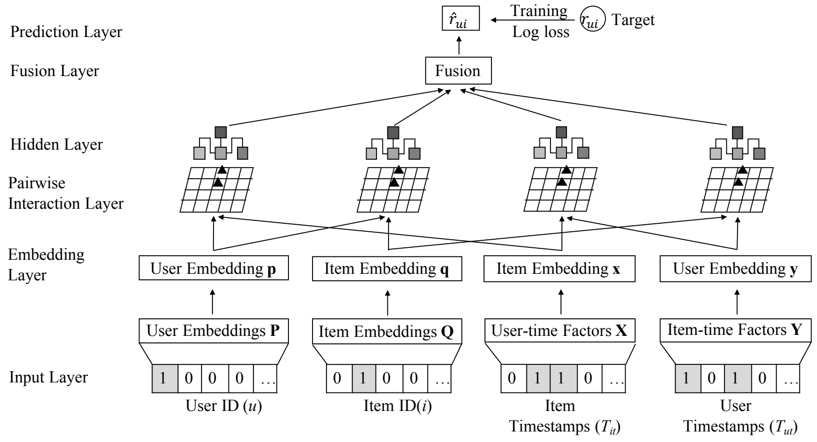


Fig. 6.2: Collective Embedding for Neural Context-aware Recommender Systems. [12]

Input Layer. The input layer encodes the user $\mathbf{u}$ and item $\mathbf{i}$ into one-hot representation vectors $\mathbf{P}$ and $\mathbf{Q}$, respectively [12]. Additionally, CoNCARS combines the binary interaction vector $\mathbf{X}$ for the observed interaction of user $\mathbf{u}$ in time $\mathbf{t}$ and the binary interaction vector $\mathbf{Y}$ for the observed interaction of item **i** in time **t**. The output of the interaction layer is four feature vectors for both **u** and **i**.

Collective Embedding Layer. The embedding layer collectively embeds the data from the input layer: user **u**, item **i**, and their observed interactions in specific time **t**. Equations 6.1 and 6.2 formally defines the user and item feature vector as an embedding vector [12].

$$\mathbf{I}_{p_u} = \mathbf{P}^T\mathbf{u} \tag{6.1}$$

$$\mathbf{I}_{q_i} = \mathbf{Q}^T\mathbf{i} \tag{6.2}$$

Moreover, the interaction vectors **X** and **Y** are denoted as latent factors associated with temporal dynamics $\mathbf{x}_u$ and $\mathbf{y}_j$, respectively. The formal definition denotes that the user **u** interacted with item **i** in different times **t** is described in Equation 6.3 and 6.4 [12].

$$\mathbf{I}_{y_u} = \frac{\sum_{i \in T(u)} \mathbf{f}_i}{\sqrt{|T(u)|}} \tag{6.3}$$

$$\mathbf{I}_{x_i} = \frac{\sum_{u \in T(i)} \mathbf{g}_u}{\sqrt{|T(i)|}} \tag{6.4}$$

where $\mathbf{I}_{y_u}$ and $\mathbf{I}_{x_i}$ are the item-based user embedding and the user-based item embedding, respectively. $\mathbf{f}_i$ is latent factors of $\mathbf{I}_{y_u}$ and $\mathbf{g}_u$ is the latent factors of $\mathbf{x}_i$. Moreover, $\mathbf{T}(\mathbf{u})$ denotes all positive entries in **X** and $\mathbf{T}(\mathbf{i})$ denotes the collection of all positive entries in **Y**. Considering the distinct items may interfer in different user's preferences, we rewrite the time-based user-item interaction as shown in Equation 6.5 and 6.6 [12].

$$\mathbf{I}_{y_u} = \sum_{i \in T(u)} \alpha_i \mathbf{f}_i \tag{6.5}$$

$$\mathbf{I}_{x_i} = \sum_{u \in T(i)} \alpha_u \mathbf{g}_u \tag{6.6}$$

where α_i is the importance degree for item **i** rated by the user **u** at time **t** and α_u is the importance degree for user **u** interaction with item **i** at time **t**. Equation 6.7 and 6.8 define the importance degree for item **i** [12].

$$\mathbf{h}_i = tanh(\mathbf{W_c}\mathbf{f}_i + \mathbf{b}_a) \tag{6.7}$$

$$\alpha_i = \frac{exp(\mathbf{h}_i^T\mathbf{h}_a)}{\sum_{i \in T(u)} exp(\mathbf{h}_i^T\mathbf{h}_a)} \tag{6.8}$$

where $\mathbf{W_c}$, $\mathbf{b}_a$, and $\mathbf{h}_a$ denote the weight matrix, bias vector, and time contextual factor, respectively. The input $\mathbf{f}_i$ for the convolutional layer produces the latent representation $\mathbf{h}_i$. Later, we compute the similarity between $\mathbf{h}_i$ and $\mathbf{h}_a$ [12]. We adopt a similar approach shown in Equation 6.7 and 6.8 to compute the time-based user representation $\mathbf{I}_{x_i}$ as explained at [12].

Pairwise Interaction Layer. The pairwise interaction layer captures the interactions between user **u** and item **i** in time **t**, which is defined in Equation 6.11 [12].

$$\mathbf{e}^n = \phi_L^n(...(\phi_2^n(z_0[n]))...) \tag{6.9}$$

$$\phi_l^n = \sigma_l^n(\mathbf{W}_l^n \mathbf{z}_{l-1}^n + \mathbf{b}_l^n), \quad l \in [1, L] \tag{6.10}$$

$$\mathbf{z}_0 = [\mathbf{p}_u \otimes \mathbf{x}_i, \mathbf{p}_u \otimes \mathbf{q}_i, \mathbf{y}_u \otimes \mathbf{x}_i, \mathbf{y}_u \otimes \mathbf{q}_i] \tag{6.11}$$

where $n \in \{1, 2, 3, 4\}$; $\mathbf{e}^n$ is the deep representation of embedding interaction learned by the n-th layer in the CNN, and $\mathbf{z}_0$ includes pairwise concatenations of user and item collective embedding as explained by Costa *et al.* [12].

Hidden Layer. The hidden layer is responsible for learning non-linear interactions from the interaction layer. The CNN is modeled as $\mathbf{c} = F_\Theta(\mathbf{I})$, where F_Θ is the model of hidden layers with parameters $\mathbf{\Theta}$, and $\mathbf{c}$ is the feature map vector used to predict the final score [12].

Fusion Layer. The hidden layer combines four latent factors vectors into a single one as formally in Equation 6.12 [12].

$$\begin{aligned} \mathbf{c}_f &= \delta_f(\mathbf{W}_f \mathbf{z}_f + \mathbf{b}_f), \\ \mathbf{z}_f &= \mathbf{c}^1 \oplus \mathbf{c}^2 \oplus \mathbf{c}^3 \oplus \mathbf{c}^4 \end{aligned} \tag{6.12}$$

where $\mathbf{W}_f$ is the weight matrix, $\mathbf{b}_f$ is the bias vector, δ_f is the activation function, and $\mathbf{z}_f$ is the concatenation of four latent interaction representations.

Prediction Layer. The prediction layer calculates the predicted score based on the assuption that user **u** prefers item **j** over item **k**. First we consider $\hat{r}_{vit} = \mathbf{w}^T \mathbf{z}_f$, where vector **w** denotes the weights for the user-item-time interactions in $\mathbf{z}_f$. Then, CoNCARS calculates the final score for user **v** and item **j** in time **t**, resulting in a tuple $(\mathbf{v}, \mathbf{j}, \mathbf{t})$. A positive result of the tuple $(\mathbf{v}, \mathbf{k}, \mathbf{t})$ denotes that $\hat{r}_{vjt}$ should be larger than $\hat{r}_{vkt}$ to have the correct label of

	Movielens				Yelp				Pinterest			
	HR@N		NDCG@N		HR@N		NDCG@N		HR@N		NDCG@N	
	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20
CAMF	0.4816	0.4929	0.1999	0.2517	0.0535	0.1006	0.0514	0.0591	0.5738	0.5911	0.3337	0.3619
TF	0.5395	0.5548	0.3076	0.4189	0.0991	0.2061	0.0953	0.0982	0.6914	0.7028	0.4574	0.4777
CHNMF	0.5500	0.5711	0.3244	0.4365	0.1121	0.2150	0.1001	0.1013	0.7164	0.7502	0.4918	0.5009
BPR	0.5841	0.6573	0.3664	0.4395	0.1558	0.2607	0.1042	0.1379	0.7464	0.8025	0.5119	0.5371
NeuMF	0.6774	0.7300	0.4133	0.4470	0.1841	0.2967	0.1095	0.1538	0.7593	0.8770	0.5324	0.5520
ConvMF	0.6801	0.7558	0.4171	0.4494	0.1937	0.3005	0.1102	0.1547	0.7921	0.8995	0.5383	0.5636
CoNCARS	**0.6974**	**0.8422**	**0.4235**	**0.4596**	**0.2442**	**0.3751**	**0.1220**	**0.1588**	**0.8801**	**0.9691**	**0.5588**	**0.5749**

Table 6.1: Top-N recommendation performance at $N = 10$ and $N = 20$. The bold font indicates the best results [12].

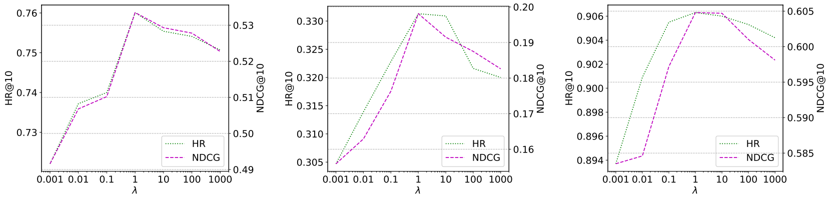


Fig. 6.3: Performance of CoNCARS regarding to hyper-parameter λ on Movielens (left), Yelp (center), and Pinterest (right) [12].

$+1$. On the other hand, a negative result of the tuple $(\mathbf{v}, \mathbf{k}, \mathbf{t})$ has a label of 0 [12].

3 Discussion

We measured the performance of CoNCARS on three benchmark datasets: Movielens, Yelp, and Pinterest. The accuracy is computed based on NDCG and HR scores to measure the quality of top-N recommendations. To do so, we performed an adapted version of the leave-one-out evaluation protocol [6, 20, 36]. The results were compared with six state-of-art approaches: CAMF [4], TF [23], CHNMF [8], BPR [31], NeuMF [20], and ConvMF [24]. Table 6.1 presents the results from the methods, where we observe CoNCARS outperforms the state-of-art methods in all datasets for N equal to 10 and 20. CoNCARS presents the best performance due to it considers the temporal features to its prediction model. Additionally, CoNCARS applies pairwise learning to capture correlations among user-item-time interactions. Moreover, CoNCARS learns non-linearities through the Convolutional Neural Network (CNN).

Additional experiments consider the variance in the results for different hyper-parameter values as illustrated in Figure 6.3 and 6.4. λ defines the regularization term, and $\mathbf{c}$ denotes the number of embedding factors per convolutional layer. The results present values of λ larger than 1 keeps the performance stable until a value of $1,000$. Considering the number of factors,

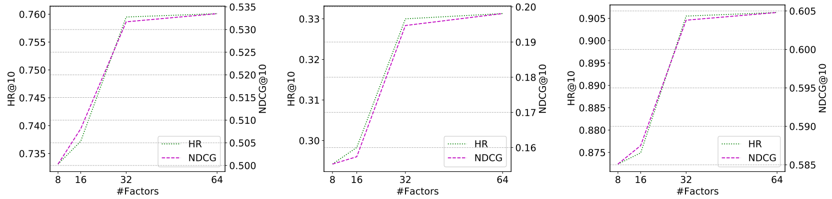


Fig. 6.4: Performance of CoNCARS regarding the number of factors on Movielens (left), Yelp (center), and Pinterest (right) [12].

CoNCARS achieves better performance when we set **c** equals to 32 and 64 in all datasets. Therefore, setting a high number of embedding factors gives a better performance as described by Costa *et al.* [12].

Convolutional Adversarial Latent Factor Model for Recommender System

This Chapter gives an overview of Paper F [11].

1 Motivation and Problem Statement

The RS models described in the previous Chapters may be susceptible to adversarial perturbations. Adversarial training has shown significant improvement in various domains, for example, image processing and Natural Language Processing (NLP). Wang *et al.* [35] and He *et al.* [19] have claimed the importance of adversarial training on recommendations to provide a more robust prediction model. Nonetheless, adversarial training applied to the implicit data face two challenges: 1) misinterpretation of null values from implicit data; and 2) unstable learning convergence. These challenges are caused by the high sparsity from implicit data and discrete values from item recommendation. To tackle these research problems, Paper F proposed by Costa *et al.* [11] describes a Convolutional Adversarial Latent Factor Model (CALF). CALF is a generative and discriminative model, where the generative is responsible for minimizing the estimated distance by capturing the true distribution, and the discriminative is responsible for estimating the distance between the generative model and the actual user preferences. We first assume the users prefer observed items over unobserved items. Then CALF utilizes the pairwise product to identify the user-item interactions. Subsequently, the output from the pairwise product becomes the input for the CNN, which is responsible for learning the non-linear user-item correlations. Lastly, the sampling method Rao-Blackwell is applied to tackle the discrete value challenge from item recommendation. CALF considers the recommendation process as a battle, where the generative model aims to improve its prediction by fighting against the discriminative model aiming to maximize the accuracy and the stability in the training step. We formally define the problem statement as [11]: *Given the generator distributions $\mathbf{s}_\theta$ and the true users's preferences $\mathbf{s}_{real}$, predict a list of items with size N for user $\mathbf{v}$ by minimizing the distance between $\mathbf{s}_\theta$ and $\mathbf{s}_{real}$. Furthermore, discriminate whether the user $\mathbf{v}$ prefers item $\mathbf{j}$ over $\mathbf{k}$ using the discriminator.*

2 Convolutional Adversarial Latent Factor Model

Figure 7.1 illustrates the CALF architecture, where we observe the generative and discriminative model. The generator is formally defined by $\mathbf{g}_\theta$, where θ is the parameter determining the generative model. On the other hand, the discriminator is formally defined by $\mathbf{d}_\phi$, where ϕ is the parameter defining the discriminative model.

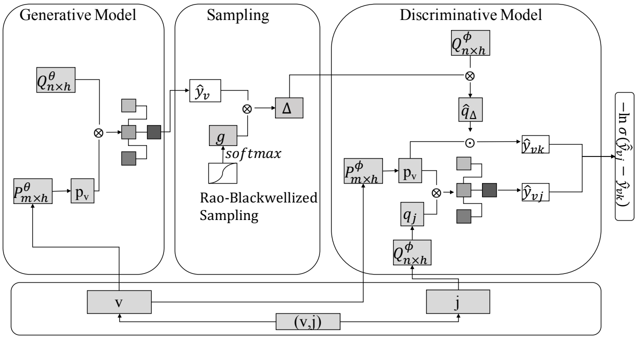


Fig. 7.1: Convolutional Adversarial Latent Factor Model [8]

The generator and discriminator are composed by a CNN responsible to learn non-linear user-item interactions. The CNN has an embedding size for the input layer of 64×64. Moreover, the channel is defined by 6 hidden layers with 32 feature maps in each layer. The feature map $\mathbf{y}$ in the hidden layer $\mathbf{l}$ is denoted by a $2D$ matrix in the interaction layer $\mathbf{S}^{ly}$. We set the stride set as $[1, 2, 2, 1]$ of the internal architecture of CNN as suggested by He *et al.* [18]. The stride denotes the *example*, *height*, *width*, and *channel* of CNN. Assuming the aforementioned definitions, the size of $\mathbf{S}^{lc}$ is half of its previous layer $l - 1$ [11]. The model functions of generator and discriminator are defined as $\mathbf{y} = F_{(\mathbf{g}_\theta|\mathbf{d}_\phi)}$, where $F_{(\mathbf{g}_\theta|\mathbf{d}_\phi)}$ is the model function with the parameters θ and ϕ [11]. Note that $\mathbf{y}$ is used to predict the final score.

Assuming the training step, the discriminatior is denoted by a tuple $(\mathbf{u}, \mathbf{j}, \mathbf{k},$ where user $\mathbf{u} \in U$, item $\mathbf{j} \in I_u^+$ and item $\mathbf{k}$ are sampled from $\mathbf{s}_\theta(k|u)$, we define the discriminator objective function as propsed by Costa *et al.* [11]:

$$J(\mathbf{g}_\theta, \mathbf{s}_\phi) = \max_\theta \min_\phi \sum_{u=1}^{m} \mathbb{E}_{j \sim \mathbf{s}_{real}(j|u) \&\& k \sim \mathbf{s}_\theta(k|u)} \ln \sigma(\hat{\mathbf{y}}_{vj} - \hat{\mathbf{y}}_{vk}), \tag{7.1}$$

where $\ln \sigma(\cdot)$ is the pairwise loss function.

CALF adopts a sampling strategy named Rao-Blackwellization [27] to minimize the issues generated by the discrete values. The sampling strategy is proposed as an alternative to policy gradient due to the instability to train and slow convergence of policy gradient. Once Rao-Blackwell sampling is applied, CALF is optimized via the gradient descent [11]. We define a vector $\hat{\mathbf{v}}_u \in \mathbb{R}^n$ to denote the item ranking scores for each user. Furthermore, a vector $\mathbf{g}_\Delta$ denotes the elements randomly resulted from the Rao-Blackwell sampling strategy $(0, 1)$. The definition is described in Equation 7.2 [11].

$$\boldsymbol{\Delta} = \frac{1}{n} \sum_{i=1}^{n} \frac{2\hat{\mathbf{v}}_u + \mathbf{g}_{\boldsymbol{\Delta}}}{g_i + 1}, \tag{7.2}$$

where $\boldsymbol{\Delta}$ is named *fake item* and is generated analogously to one-hot item vector. A differentiable method to define the feature vector for each *fake item* is necessary, because the generator may generate an infinite number of *fake items*. The differentiable method is formulated as in Equation 7.3 as defined by Costa *et al.* [11].

$$\hat{\mathbf{q}} = \boldsymbol{\Delta}\mathbf{Q} \tag{7.3}$$

$\mathbf{Q} \in \mathbb{R}^{n \times h}$ denotes the *fake items* embedding matrix, $\hat{\mathbf{q}} \in \mathbb{R}^h$ denotes the feature vector of item $\boldsymbol{\Delta}$, and h represents the number of latent features [11].

CALF solves research problem of the discrete values and facilitates the gradient updates in the generator by applying the strategy mentioned above.

Subsequently, CALF updates the parameters ϕ from the discriminator utilizing the gradient descent to minimize the objective function as formulated in Equation 7.4 [11].

$$\phi \leftarrow \phi - lr \times \nabla_\phi \ln \sigma(\hat{\mathbf{y}}_\phi(v_j) - \hat{\mathbf{y}}_\phi(v_k)) \tag{7.4}$$

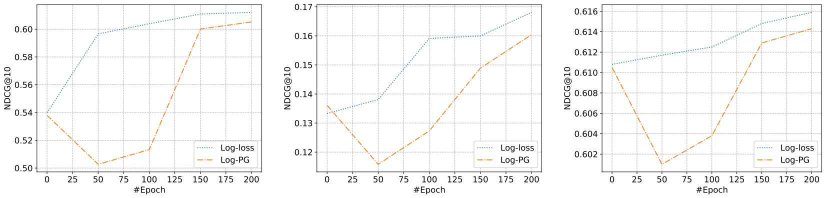
On the other hand, the parameters θ from the generator is updated utilizing the gradient ascent to maximize the objective function as defined in Equation 7.5 [11].

$$\theta \leftarrow \theta + lr \times \nabla_\theta \ln \sigma(\hat{\mathbf{y}}_\theta(v_j) - \hat{\mathbf{y}}_\theta(v_k)), \tag{7.5}$$

where lr denotes the learning rate.

	Movielens				Yelp				Pinterest				
	HR@N		NDCG@N		HR@N		NDCG@N		HR@N		NDCG@N		RI
	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	
AMF	0.5331	0.7255	0.3517	0.4444	0.1176	0.2385	0.0950	0.1065	0.7098	0.8972	0.4946	0.5658	+31%
IRGAN	0.5400	0.7301	0.3744	0.4665	0.1321	0.2550	0.1035	0.1113	0.7200	0.9002	0.5111	0.5832	+25%
CNCF	0.6103	0.8041	0.4316	0.5011	0.1578	0.2686	0.1073	0.1200	0.7489	0.9026	0.5367	0.5881	+20%
CFGAN	0.6805	0.8352	0.4991	0.5640	0.1829	0.2889	0.1184	0.1459	0.7668	0.9053	0.5513	0.5928	+15%
CALF	**0.7124**	**0.8596**	**0.5121**	**0.6153**	**0.2037**	**0.3148**	**0.1364**	**0.1681**	**0.7811**	**0.9155**	**0.5742**	**0.6159**	-

Table 7.1: Top-N recommendation performance at $N = 5$ and $N = 10$. The bold font indicates the best results. RI indicates the relative improvement of CALF over the corresponding baseline on average. [11]



(a) Movielens **(b)** Yelp **(c)** Pinterest

Fig. 7.2: Performance of CALF regarding differentiable sampling and policy gradient. [11]

3 Discussion

The performance of CALF is measured on three benchmark datasets: Movielens, Yelp, and Pinterest. The accuracy is computed based on NDCG and HIT scores to measure the quality of top-N recommendations. We performed an adapted version of the leave-one-out evaluation protocol [18, 20]. The results were compared with four state-of-art approaches: AMF [19], CNCF [18], IRGAN [35], and CFGAN [5]. Table 7.1 presents the results of the experiments, where we observe CALF outperforms the state-of-art methods in all datasets for N equal to 5 and 10. CALF presents the best performance in comparison with the state-of-art methods due to 1) learning non-linearities through the CNN; 2) applying pairwise product to capture the user-item correlations better; 3) applying adversarial training to refine the relevance between users and items; and 4) utilize the Rao-Blackwell sampling strategy to deal with null values and discrete values from the data.

Moreover, we analyze the benefits of applying the sampling strategy as illustrated in Figure 7.2. Figure 7.2 presents the gradient descent has a stable performance and faster convergence in comparison with policy gradient. Therefore, the sampling strategy adopted by CALF brings benefits to the model.

Finally, we measure the time convergence, and the results are shown in Table 7.2. According to the table, the runtime of spent by CALF in the generator and discriminator for each epoch is longer when compared to the other

	Movielens					**Yelp**					**Pinterest**				
	CALF	**AMF**	**CNCF**	**IRGAN**	**CFGAN**	**CALF**	**AMF**	**CNCF**	**IRGAN**	**CFGAN**	**CALF**	**AMF**	**CNCF**	**IRGAN**	**CFGAN**
D	1 m	-	-	45 s	50 s	1.7 m	-	-	55 s	1.3 m	1.9 m	-	-	1.9 m	1.9 m
G	1.7 m	-	-	1.5 m	1.2 m	3.9 m	-	-	1.7 m	2.5 m	3.7 m	-	-	4.9 m	3.9
EC	50	60	50	60	60	50	100	90	120	100	70	90	80	70	70
TC	3 h	4 h	3.5 h	5 h	4.6 h	4.6 h	5.3 h	5 h	7 h	6.5 h	6.5 h	6.9	6.8 h	8 h	7 h

Table 7.2: Convergence time. - denotes the methods without generative and discriminative models; D is the discriminative model; G is the generative model; EC denotes the epoch convergence; and TC denotes the time convergence [11]

models. Nevertheless, the overall training time is lower than the other models and thereby improving the computational time.

Summary of Contributions

The thesis provides a multi-view LFM to improve predictions in RS using explicit and implicit user's feedback. The thesis is composed of 6 papers., which has the contributions summarized below.

- Paper A [8] introduces CHNMF as a hybrid recommendation model to provide better predictions. First, the paper describes which features affect the prediction model. CHNMF utilizes collective non-negative matrix factorization by combining two learning methods, Barzilai-Borwein optimization, and multiplicative update rules. The empirical evaluation presents CHNMF outperform the state-of-art methods in precision, recall, f-measure, and NDCG. Moreover, CHNMF presents a faster learning convergence in denser datasets.

- Paper B [32] exploits different features given by LBSN to provide the CMViP model for predicting visitors at a given location and a particular time. The paper introduces the features which may affect the mobility of users and the algorithm to compute the probability of a user visiting a target location based on those features. Initially, CMViP applies a CMF model to identify the user's preferences, then combine the outputs with the influence maximization method. The experimental evaluation presents CMViP outperforms the state-of-art methods in both precision and recall.

- Paper C [9] proposes a review-based text generation model to explain recommendations based on the user's preferences. The paper describes the LSTM network applied in the model, as well as, the attention layer which generates text based on the user's previous review style. The experiment shows the model's ability to provide a review based on the predicted ratings. Moreover, the generated text provides good readability based on the performed experimental results.

- Paper D [10] extends paper A in two directions: by adding sentiment as additional information to the model's prediction and proposing an explainable method to LFM. The paper introduces the NECoNMF model, which collectively factorizes ratings, content features, sentiment, and contextual information in a common latent space. Then, the paper describes the explainable neural model utilized to interpret the predicted top-N recommendation based on the LFM. The results provided by the experiments show that NECoNMF outperforms the state-of-art methods in both HR and NDCG.

- Paper E [12] describes CoNCARS, a model to provide item recommendations using implicit data. The paper introduces the importance of time as a feature in predicting users preferences. Due to the assumption that users may repeat their actions in the future. CoNCARS combines the users, items, and time embedding features using LFM. Subsequently, CoNCARS applies CNN to learn non-linear correlations among the features to predict the user's preferences. The empirical evaluation presents CoNCARS improves the top-N recommendation in both HR and NDCG in comparison with the state-of-art methods. Furthermore, an experimental evaluation was performed to present CoNCARS performance for different hyper-parameter values.
- Paper F [11] proposes an adversarial training for the CALF LFM model, to providing predictions in RS. The paper introduces the generative and discriminative models to refine the relevance between the user and items correlations. Furthermore, CALF utilizes the Rao-Blackwell sampling strategy to deal with the discrete values problem, providing a faster and more stable learning convergence. The empirical evaluation presents CALF outperforms the state-of-art methods in both HR and NDCG for top-N items' recommendation in comparison with the state-of-art techniques. Moreover, CALF presents a more stable training curve and spends less computational time during the training step.

The thesis addresses the following research problems: sparsity, explainability, and null values in RS. Paper A, Paper B, and Paper E focus on utilizing contextual information to infer top-N recommendations based on the users' preferences and improve the recommendations in the sparse domains. Paper A and Paper B apply CMF to identify linear correlations of user-item interactions. Paper B extends Paper A by defining a new research problem and incorporate information into the contextual features. Paper E propose a neural model to improve the models introduced in Paper A and Paper B. The model utilizes a CNN layer to learn non-linearities from users and items correlations considering time.

Paper C and Paper D addresses the lack of explainability as a research problem in RS. Paper C explores a new interpretable model based on the neural network, which primarily trains an LSTM network based on previous users' reviews and ratings. The personalized text is generated based on the output of the LSTM network and an attention layer. The model proposed in Paper D extends Paper A by adding the sentiment to its factorization. Based on the factorized features and the explainable model proposed in Paper C, NECoNMF can explain the recommendations with unambiguous interpretability.

The models described above may be susceptible to adversarial perturbations. To improve the recommendation given the null values and discrete values, Paper F introduces the CALF model, which applies adversarial training utilizing the Rao-Blackwell sampling strategy. The proposed solution improves the results in comparison with Paper E by dealing with discrete items, improving the stability and computational time convergence to the model.

Detailed information about all the contributions can be found in Part II of this thesis or directly in the cited research papers.

Future Directions

The contributions of the thesis can be applied to increase the users' satisfaction given by the RS. Future generations of recommendation models may utilize the proposed methods to evaluate different hypothesis regarding context, explainability, and other research challenges given by explicit and implicit data. For instance, a recommended list of item's suggested to a user during a weekday might not be similar to the list suggested on the weekend. Another example would be that the RS needs to explain *why* the recommended list of items is similar to the user's previous preferences. Our proposed methods to recommend top-N items will provide an understanding of such user-item interactions.

The models proposed in this thesis enable extensions in different directions. Paper A could exploit the cold-start research problem in RS to improve the efficiency of the proposed method. Paper B might, for example, add features for optimizing predictions, such as weekdays or weekend and popularity of locations. Paper C can explore advanced techniques in the NLP research field to improve the readability of the text and consider the review's usefulness score, which may influence the quality of the explanation. Paper D may exploit the user's social relations to improve the accuracy of the proposed model. Paper E could consider sequential-based information to the models since users may change their preferences in the near future. Therefore, the sequence becomes an essential feature for the online systems scenario. The method proposed in Paper F can investigate the influence of richer contexts, such as the reviews given by a user.

The proposed solutions present how the research can be applied in the RS field, and the future directions present how practitioners improve the proposed methods. Moreover, future directions relate to modeling users and items to identify the demands of real-world online systems. The suggestions raised by this Ph.D. thesis opens up a wide range of possibilities for the satisfaction of users on the Web.

References

[1] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recc Sys. handbook*. Springer, 2011, pp. 217–253.

[2] J. Anderson, "Lix and rix: Variations on a little-known readability index," *Journal of Reading*, vol. 26, no. 6, pp. 490–496, 1983.

[3] I. Avazpour, T. Pitakrat, L. Grunske, and J. Grundy, *Dimensions and Metrics for Evaluating Recommendation Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 245–273.

[4] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, "Incarmusic: Context-aware music recommendations in a car," in *E-Commerce and Web Technologies*. Springer, 2011, pp. 89–100.

[5] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J.-T. Lee, "Cfgan: A generic collaborative filtering framework based on generative adversarial networks," in *CIKM*, 2018, pp. 137–146.

[6] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, "Delf: A dual-embedding based deep latent factor model for recommendation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3329–3335. [Online]. Available: https://doi.org/10.24963/ijcai.2018/462

[7] M. Coleman and T. L. Liau, "A computer readability formula designed for machine scoring." *Journal of Applied Psychology*, vol. 60, no. 2, p. 283, 1975.

[8] F. Costa and P. Dolog, "Hybrid learning model with barzilai-borwein optimization for context-aware recommendations," in *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida USA., May 21-23 2018.*, 2018, pp. 456–461.

[9] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations," in *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, ser. IUI'18. ACM, 2018, pp. 57:1–57:2.

[10] F. Costa and P. Dolog, "Neural explainable collective non-negative matrix factorization for recommender systems," in *Proceedings of the 14th International Conference on Web Information Systems and Technologies - Volume 1: WEBIST,*, INSTICC. SciTePress, 2018, pp. 35–45.

[11] F. Costa and P. Dolog, "Convolutional adversarial latent factor model for recommender system," in *Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference (Submitted)*, 2019.

[12] F. Costa and P. Dolog, "Collective embedding for neural context-aware recommender systems," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization (Submitted)*, 2019.

[13] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 39–46.

[14] K. Eriksson, D. Estep, and C. Johnson, *Lipschitz Continuity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 149–164.

[15] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors." in *AAAI*, 2017.

[16] R. Gunning, "The technique of clear writing," 1952.

[17] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1661–1670.

[18] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*, 2018, pp. 2227–2233.

[19] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial personalized ranking for recommendation," in *SIGIR*, 2018, pp. 355–364.

[20] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW '17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[21] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, Jan. 2004.

[22] Y. Huang, H. Liu, and S. Zhou, "An efficient monotone projected barzilai–borwein method for nonnegative matrix factorization," *Applied Mathematics Letters*, vol. 45, pp. 12–17, 2015.

[23] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/1864708.1864727

[24] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16. New York, NY, USA: ACM, 2016, pp. 233–240.

[25] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," 1975.

[26] Y. 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*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401944

[27] R. Liu, J. Regier, N. Tripuraneni, M. I. Jordan, and J. McAuliffe, "Rao-blackwellized stochastic gradients for discrete distributions," *CoRR*, 2018.

[28] S. Maharjan, J. Arevalo, M. Montes, F. A. González, and T. Solorio, "A multi-task approach to predict likability of books," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, ser. EACL'17, vol. 1, 2017, pp. 1217–1227.

[29] G. H. Mc Laughlin, "Smog grading-a new readability formula," *Journal of reading*, vol. 12, no. 8, pp. 639–646, 1969.

[30] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," in *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, 1998, pp. 161–172.

[31] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09. Arlington, Virginia, United States: AUAI Press, 2009, pp. 452–461. [Online]. Available: http://dl.acm.org/citation.cfm?id=1795114.1795167

[32] M. A. Saleem, F. S. da Costa, P. Dolog, P. Karras, T. Calders, and T. B. Pedersen, "Predicting visitors using location-based social networks," in *MDM*, 2018, pp. 245–250.

[33] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: ACM, 2014, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/2645710.2645751

[34] R. Senter and E. A. Smith, "Automated readability index," CINCINNATI UNIV OH, Tech. Rep., 1967.

[35] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *SIGIR*, 2017, pp. 515–524.

[36] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 3203–3209. [Online]. Available: https://doi.org/10.24963/ijcai.2017/447

[37] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, ser. SIGIR '14. New York, NY, USA: ACM, 2014, pp. 83–92.

Part II

Papers

Paper A

Hybrid Learning Model with Barzilai-Borwein Optimization for Context-aware Recommendations

Felipe Costa and Peter Dolog

The paper has been published in the
Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS'18), pp. 456–461, 2018.

Abstract

We propose an improved learning model for non-negative matrix factorization in the context-aware recommendation. We extend the collective non-negative matrix factorization through hybrid regularization method by combining multiplicative update rules with Barzilai-Borwein optimization. This provides new improved way of learning factorized matrices. We combine ratings, content features, and contextual information in three different 2-dimensional matrices. We study the performance of the proposed method on recommending top-N items. The method was empirically tested on 4 datasets, including movies, music, and mobile apps, showing an improvement in comparison with other state-of-the-art for top-N recommendations, and time convergence to the stationary point for larger datasets.

1 Introduction

Recommender systems are traditionally focused on users, items, and their interactions to build a model to recommend a sorted list of N items, corresponding to the user's interests. However, it is important to incorporate context in some applications during the recommendation process, such as tourism (sights to be visited), movies (time and place), and so on. Researchers have identified the quality of recommendations increases when they use additional information, such as *time* and *location* [1].

There are two main challenges in recommendation process: 1) generating list of top-N recommendations and 2) time of convergence in learning the factorized matrices. Context-aware recommender models has shown significant improvement to cover the first challenge as presented by [2–4], however, their models do not consider content features, which may influence the users' decision and improve recommendation accuracy. For the second challenge ALS has been applied in different matrix factorization models as presented by [5, 6]. Nonetheless, the convergence results for gradient descent methods assume the subproblems have unique solutions [7].

As a solution for top-N recommendations and convergence of learning curve, we propose to extend the collective non-negative matrix factorization (CNMF) using the Barzilai-Borwein (BB) optimization method and multiplicative update rules, called the collective hybrid non-negative matrix factorization (CHNMF[1]). CHNMF factorizes ratings, content features and context in three non-negative low-rank matrices, represented in a common latent space. Our hypothesis is that using the same factor space to jointly decompose different matrices (e.g. attributes, context, and users' tastes) improves the prediction of top-N items. Further, BB improves convergence time for the learning model, by running the factorization tasks in parallel. Factorizing the rating, content features and contextual information collectively allows BB to perform better in larger dataset than ALS, due to higher density. Hence, BB only computes two projections and two gradients at even steps. Moreover, it determines the step length without using any line search.

We performed an experimental evaluation of the models on 4 datasets: LDOS-CoMoDa, InCarMusic, Frappe, and Movielens. The proposed model outperforms the state-of-art regarding convergence time in learning, as well as, in accuracy measured by metrics commonly used for evaluation of top-N recommendations quality.

This paper has the following contributions:

- An efficient hybrid learning model, based on multiplicative update rules and Barzilai-Borwein optimization;

[1] Available at: `https://github.com/felipeecosta`

- A collective model combining ratings, content features, and contextual information into a collective hybrid non-negative matrix factorization framework;
- Empirical experiments comparing the results between CHNMF and state-of-the-art methods for top-N recommendation.

2 Related Works

Related work can be divided in two areas: context-aware recommender systems, and collective matrix factorization (CMF).

Context-aware Recommender Systems. [1] categorized context-aware recommender systems (CARS) into: pre-filtering, post-filtering, and contextual modeling. In this paper we have not considered post-filtering approach as baseline, because it has shown less efficiency in comparison with the others [3].

Pre-filtering. [2] introduces the UI-Splitting approach, which splits a given rating vector into two virtual vectors using a specific contextual factor. [3] presents the distributional-semantics pre-filtering (DSPF), which proposes to build a matrix factorization using classified ratings with the most similar contextual situations.

Contextual modeling. [4] proposes the context-aware matrix factorization (CAMF), which extends matrix factorization using context as baseline predictor to represent interaction of contextual information with items or users. [8] discuss contextual SLIM (CSLIM) technique, which incorporates contextual factor to SLIM algorithm, through estimating the ranking score $\hat{S}_i, j, c$ for user u_i in item t_j in context c.

Collective Matrix Factorization. Multi-view clustering is a technique to split objects into clusters based on multiple representations of the object. [5, 6] propose different methods using CMF. [5] proposed the MultiNMF, using a connection between NMF and PLSA. Comparing different views of factors in multi-view setting for clustering. [6] proposed a co-regularized NMF (CoNMF), where comment-based clustering is formalized as a multi-view problem using pair-wise and cluster-wise CoNMF.

Decoupled Target Specific Features Multi-Target Factorization (DMF), proposed by [9], follows the same principle of CMF. DMF learns a set of single target models optimized for one relation, while downweighting the others. However, a number of parameters are used only for auxiliary relations and never for predicting the targets, what diverges from the proposed work in this paper.

Local collective embeddings (LCE) is a matrix factorization method proposed by [10], which exploits user-document and document-terms matrices, identifying a common latent space to both item features and rating matrix. LCE has shown effectiveness in cold-start problem for news recommendation, however, it has some limitations. The method does not perform well in our domain which covers movies, music and mobile apps, because it uses only two matrices as input and multiplicative update rules as learning model. In this paper, we extend the LCE approach proposing CHNMF to address these limitations. CHNMF decomposes a matrix as a product of three matrices: content features, rating, and context. Content features are data from each item's metadata, ratings represents user's preferences, while contextual information is the situation where the user rates an item. Furthermore, the hybrid technique is applied using multiplicative update rules and Barzilai-Borwein optimization to provide a faster convergence to stationary point during the learning model.

3 Problem Formulation

The research problem investigated in this paper is defined as follows: *Recommend a ranked list of items to each user, given by ratings, content features, and contextual information on user-item interactions*. Modeling the rating data from U users to I items under X_a types of content and X_c types of context as three 2-dimensional matrices, i.e., user-item matrix as $X_u \in \mathbb{R}^{u \times i}$; user-content feature matrix is formally defined as $X_a \in \mathbb{R}^{u \times a}$; and user-context matrix as $X_c \in \mathbb{R}^{u \times c}$. Where, u is the number of users, i is the number of items, a is the content size, and c is the context a user rated an item. The matrix $X_a \in \{0, 1\}$ represents whether a target preferable item belongs to a specific attribute or not. The rating matrix presents the user's preferences in a numerical scale as $X_u \in \{1, 2, 3, 4, 5\}$. Finally, $X_c \in [0, 1]$ presents how often a user rated an item in a specific context.

Factor models aims to decompose the original user-item interaction matrix into two low-rank approximation matrices. CHNMF is a generalization of the classical matrix factorization methods for content features and contextual information. The latent features are stored in three low-rank matrices: ratings as $W \times H_u$; categories as $W \times H_a$; and context $W \times H_c$. Where, H_u denotes a row vector, which represents the latent features for user u. Similarly, H_a represents the category's latent features a, and H_c represents the context's latent features c.

4 Collective Non-negative Matrix Factorization

Considering the notation used in the problem formulation, it is factorized X_a into two lower dimensional matrices, obtaining the $factor \times attributes$ scores belonging to an item. Factorizing X_u leads to find $factor \times items$ scores, presenting the users' preferences. Likewise, the factorization of matrix X_c allow us to identify the hidden contextual factors related to the item. CHNMF represents ratings, content features, and contexts in a common latent space, collectively factorizing X_u, X_a, and X_c into a low-dimensional representation. The formal definition, is given as the following optimization problem:

$$\begin{aligned} min: f(W) = \frac{1}{2}[\alpha \| X_u - WH_u \|_2^2 + \beta \| X_a - WH_a \|_2^2 \\ + \gamma \| X_c - WH_c \|_2^2 \\ + \lambda(\| W \|_2 + \| H_u \|_2 + \| H_a \|_2 + \| H_c \|_2)] \\ s.t. W \geq 0, H_u \geq 0, H_a \geq 0, H_c \geq 0 \end{aligned} \tag{A.1}$$

where W represents the common latent space during the decomposition of X_u, X_a, and X_c. $\{\alpha, \beta, \gamma\} \in [0,1]$ are hyper-parameters controlling the importance of each factorization. The remaining terms are Tikhonov regularization of W, H_u, H_a, and H_c controlled by the hyper-parameter $\lambda \geq 0$, used to enforce a smooth solution and avoid overfitting.

4.1 Optimization

The optimization performs as follows: (1) fix the value of W while minimizing $f(W)$ over H_u, H_a, H_c; then (2) fix the value of H_u, H_a, H_c while minimizing $f(W)$ over W. Considering a matrix with x_i rows and y_j columns, with a relation defined by r_{ij}, we can define the correlation among n neighbors' data points. This results in a matrix A, used to measure the local closeness of two data points x_i and y_j.

Collective factorization reduces data points x_i from a matrix X, into a common-latent space W as w_i. The distance between two low dimensional data points is calculated using the Euclidean distance: $\|w_i - w_j\|^2$, and mapped into a matrix A. Based on the matrix A we can iterative run these two steps until the stationary point, or until the established number of max iterations as follow:

$$\begin{aligned} M &= \frac{1}{2}\sum_{i,j=1}^{n} \|w_i - w_j\|^2 A_i j \\ &= \sum_{i=1}^{n}(w_i^T - w_i)D_i i - \sum_{i,j=1}^{n}(w_i^T - w_i)D_i i \\ &= Tr(W^T DW) - Tr(W^T AW) = Tr(W^T LW), \end{aligned} \tag{A.2}$$

where $Tr(\bullet)$ denotes the trace function, and D is a diagonal matrix whose entries are row sums of A (or column, as A is symmetric), in other words, $D_i i = \sum_i A_i j$; $L = D - A$ is called the Laplacian matrix, we need to incorporate it to enforce the non-negative constraints.

The optimization problem of function $f(W)$ is written as:

$$\begin{aligned} min : f(W) = \frac{1}{2}[\alpha\|X_u - WH_u\|_2^2 + \beta\|X_a - WH_a\|_2^2 \\ +\gamma\|X_c - WH_c\|_2^2 + \varphi Tr(W^T LW) \\ +\lambda(\|W\|_2 + \|H_u\|_2 + \|H_a\|_2 + \|H_c\|_2)] \\ s.t. W \geq 0, H_u \geq 0, H_a \geq 0, H_c \geq 0 \end{aligned} \tag{A.3}$$

where L is the Laplacian matrix, and φ is a hyper-parameter which controls the objective function.

5 Hybrid Learning Model

CHNMF is a non-convex method, considering all parameters (W, H_u, H_a, H_c) together, it is unrealistic to expect the algorithm to find the global minimum. [10] propose an iterative algorithm based on multiplicative update rules (MUR) to achieve the stationary point. However, it has been observed that MUR converges relatively slowly [7]. In this paper, we present a hybrid learning model using MUR and Barzilai-Borwein (BB) method to solve the convergence problem.

5.1 Barzilai-Borwein

Since H_u, H_a, and H_c have the same behaviour, we represent them in this paper as H. We have to solve:

$$\min_{W \geq 0} : f(W, H) = \frac{1}{2}\|X - WH\|_F^2 \tag{A.4}$$

We map all the negative values into zero through $P(.)$. As H is a stationary point of Equation A.4 for any $\alpha > 0$, then,

$$\|P[H - \alpha\nabla f(H)] - H\|_F = 0. \tag{A.5}$$

The gradient $\nabla f(W)$, of $f(H)$, is Lipschitz continuous with constant $L = \|W^T W\|_2$. Since $W^T W$ is a $k \times k$ and $k \ll min\{m, n\}$, the Lipschitz constant L is not expensive to obtain.

We use $\|P[H - \alpha\nabla f(H)] - H\|_F \leq \epsilon_H$, where $\epsilon_H = max(10^{-3}, \epsilon)\|P[H - \alpha\nabla f(H)] - H\|_F$. If Algorithm 1 solves Equation A.4 without any iterations, we decrease the stopping tolerance by $\epsilon = 0.1\epsilon_H$. For a given $H_0 \geq 0$:

$$\mathcal{L}(H_0) = \{H | f(H) \leq f(H_0), H \geq 0\}. \tag{A.6}$$

By the definition of Equation A.4 we have the stationary point of the Barzilai-Borwein method.

5.2 Multiplicative Update Rules

We combine multiple regularization methods, where H_u, H_a, H_c converge using the Barzilai-Borwein method, while W uses multiplicative update rules to achieve the stationary point. The partial derivatives of $f(W)$ is:

$$\begin{aligned} \nabla f(W) = \alpha W H_u H_u^T - \alpha X_u H_u^T + \beta W H_a H_a^T \\ -\beta X_a H_a^T + \gamma W H_c H_c^T - \gamma X_c H_c^T + \lambda I_k \end{aligned} \tag{A.7}$$

where I_k is the identity matrix with $k \times k$ dimensions. Applying the Karush-Kuhn-Tucker(KKT) first-order optimal conditions to $f(W)$, we derive:

$$W \geq 0, \nabla f(W) \geq 0, W \odot \nabla f(W) = 0, \tag{A.8}$$

where $\odot$ corresponds to the element-wise matrix multiplication operator.

Substituting the derivatives of $f(W)$ from Equation A.7 in Equation A.8 leads to following update rules:

$$W = \frac{[\alpha X_u H_u^T + \beta X_a H_a^T + \gamma X_c H_c^T]}{[\alpha H_u H_u^T + \beta H_a H_a^T + \gamma H_c H_c^T + \lambda I_k]}, \tag{A.9}$$

where $\frac{\bullet}{\bullet}$ corresponds to the element-wise matrix division.

Algorithm 1 Barzilai-Borwein

Input : feature matrix H, common latent space W, and convergence criteria.
Output: H

```
$\sigma \in (0,1)$
 $\alpha_{max} > \alpha_{min} > 0$
 $L \leftarrow \|W^T W\|_2$
 $H_0 \leftarrow H^k$
 $\alpha_0 \leftarrow 1$
 $t \leftarrow 0$
 if $H_t$ is a stationary point of A.1 then
    return $H_t$
else
    while not converged do
        if $t/2 \neq 0$ then
            return $Z_t \leftarrow H_t$
        else
            $Z_t \leftarrow P\left[H_t - \frac{1}{L}\nabla f(H)\right]$
        end
        $D_t \leftarrow P\left[Z_t - \alpha_t \nabla f(Z_t)\right] - Z_t$
         $\delta \leftarrow \langle D_t, WW^T D_t \rangle$
         if $\delta_t \leftarrow 0$ then
            return $\lambda_t \leftarrow 1$
        else
            $\lambda_t \leftarrow min\{\tilde{\lambda_t}, 1\}$
             where $\tilde{\lambda_t} \leftarrow -\frac{(1-\sigma)\langle \nabla f(Z_t), D_t \rangle}{\delta_t}$
             $H_{t+1} \leftarrow Z_t + \lambda_t D_t$
        end
        $S_t \leftarrow H_{t+1} - H_t$
         $Y_t \leftarrow \nabla f(H_{t+1}) - \nabla f(H_t)$
         if $\langle S_t, Y_t \rangle \leq 0$ then
            $\alpha_{t+1} \leftarrow \alpha_{max}$
        else
            if $t/2 \neq 0$ then
                $\alpha^{BB}_{t+1} \leftarrow \frac{\langle S_t, S_t \rangle}{\langle S_t, Y_t \rangle}$
            else
                $\alpha^{BB}_{t+1} \leftarrow \frac{\langle S_t, Y_t \rangle}{\langle Y_t, Y_t \rangle}$
            end
            $\alpha_{t+1} \leftarrow min\{\alpha_{max}, max\{\alpha_{min}, \alpha^{BB}_{t+1}\}\}$
        end
        $t \leftarrow t + 1$
    end
end
return $H$
```

Each iteration of CHNMF algorithm gives us a solution for the pair-wise division. As we map any negative values to zero, the W matrix becomes a non-negative after each update. Furthermore, the objective function and the delta decrease on each iteration of the above update rules, guaranteeing the convergence into a stationary point.

5.3 Complexity Analysis of CHNMF

[10] applied MUR using ALS as learning model due to its efficiency and simplicity. LCE updates matrix factors by multiplying each entry with a positive factor in every iteration round. However, MUR converges relatively slowly [7].

CHNMF is non-convex and NP-hard problem, in relation to the variables W and H. However, ALS optimizes the subproblems W and H into convex problems. Despite the optimization, they might have more than one optimal solution because they are not strictly convex. The convergence gradient descent method assumes the subproblems have unique solutions [7]. Furthermore, most of the methods applying ALS are inefficient in finding a step length by using the line search, resulting in a slow convergence.

Regarding the computational complexity given by multiplicative update rules, $X_u(H_u^k)^T$, $X_a(H_a^k)^T$, $X_c(H_c^k)^T$, $(W^(k+1))^T X_u$ are $O(nmr)$ operations, where n and m are the matrix dimensions, and r is the stationary point. The former operations are $O(nmr)$, but the latter costs $O(max(m,n)r^2)$. When $r < min(m,n)$ the latter is better. In summary, the overall cost of MUR is: $\#iterations \times O(nmr)$.

Monotone projected BB optimization model is used to solve CHNMF subproblems because it uses four stepsizes to improve the performance of the gradient methods [11]. Finally, it determines the step length without using any line search.

CHNMF presents its highest complexity in conditional terms described between line 12 and 16 in Algorithm 1, besides the gradient computation itself. The complexity is shown as $O(nmr) + \#sub - iterations \times O(tmr^2)$, where k is the number of features. Consider H_u, H_a, and H_c are constant matrices. The overall cost is: $\#iterations \times O(nmr)$
$+ \#sub - iterations \times O(kmr^2 + knr^2)$.

There are two $O(nmr)$ operations for each iteration: $X_u(H_u^k)^T$, $X_a(H_a^k)^T$, $X_c(H_c^k)^T$, $(W^(k+1))^T X_u$, as multiplicative update method. However, when k and $\#sub - iterations$ are small, this method is more efficient.

Big O notation aforementioned shows an improvement on the convergence when the factorization task is paralleled into two different learning processes, as small sub-threads. In this case W uses MUR method, and H_u, H_a, H_c uses Barzilai-Borwein optimization.

6 Recommendation Process

Barzilai-Borwein and multiplicative rules return the trained matrices W, H_u, H_a, and H_c containing the scores for prediction. Given the vector of unseen items v_i, we can predict the most preferable items according to the user's taste, represented as v_u. CHNMF projects the items vector v_i to the common latent space by solving the overdetermined system $v_i = wH_u$. The vector w, captures the factors to explain the preferable items v_i. Then, it uses low dimensional vector w to infer the missing part of the query: $v_u \leftarrow wH_t$. H_t is the concatenation of attribute and contextual matrices $H_t = H_a||H_c$. CHNMF ranks the items according to the predictions of the user's preference to unseen items stored in v_u.

6.1 Parameter Analysis

CHNMF has 5 essential parameters: k, the number of latent factors; α, β, and γ balance the factorization among ratings, content features, and contextual information; and λ, controlling the smoothness of the solution. The parameter k controls the quantity of factors considered by the system, consequently the complexity of the model. The small values of k underfit, while large values of k overfit the data and lead to poor performance.

Setting α, β, and γ with the same values give equal importance to all matrices, while parameters with different values give different levels of importance to each matrix. Setting the importance degree of ratings, content features and contextual information, for example, α β, and γ, ≈ 0.33 tends to achieve the best performance. Low values of α β, and γ tends to show lower performance in ranking quality.

The smoothness hyper-parameter $\lambda \geq 0$ is used to avoid overfitting. Lower values of λ oversimplify the model and decrease performance.

7 Experiments

Datasets. Four datasets are used to compare the methods: LDOS-CoMoDa [12], InCarMusic [4], Frappe [13], and Movielens [14]. We performed a t-test to analyze the datasets' statistical significance of null hypothesis H_0: "if movie A and B share the same content-features and they are frequently viewed together, there should be some hidden relationships between them that raise the user's curiosity". In the case, the datasets do not reject the null hypothesis at the significance level $\alpha = 0.05$, presenting p-value as 0.0262 (LDOS-CoMoDa), 0.0393 (inCarMusic), 0.0365 (Frappe), and 0.0348 (Movielens).

Baselines for Comparison. *Pre-filtering*. UISplitting and DSPF techniques are trained on the ratings tagged with contextual similar situations to compute rating predictions for a specific target context.

Contextual-modeling. CAMF-C, CSLIM-ICS, CSLIM-LCS, and CSLIM-MCS, had their setup defined as recommended by [4, 8].

LCE. It was defined $\alpha = 0.5$ and $\lambda \in [0, 1]$ as recommended by [10], which had a better performance in their experiments.

CoNMF. It follows the authors' suggested settings [6], where they propose the regularization parameters set to 1 for all ratings and datasets. This model was applied before the recommender process to compare the technical performance.

MultiNMF. The authors suggested to set the regularization parameters uniformly to 0.01 [5]. Initially, MultiNMF normalizes the data matrix using L1-norm, however, to become consistent with the technique presented in this paper it was decided to test it using L2-norm.

Evaluation Metrics. NDCG, precision, recall and f-measure are used to test the ranking quality based on user's preferences scores generated by CHNMF. We set $N = 5$ and $N = 10$ because this value retrieves a smaller list of items, considering the user's taste. Large values of N would result in the extra work for the user to filter among a long list of relevant items.

To avoid overfitting we perform the experiments using 5-fold cross validation.

Results. The experiments were performed on Unix server with 32GB of RAM and 8 core CPU Intel Xeon with 2.80GHz, under the same parameters settings: *learning rate* = 0.001; *k* = 50; *iterations* = 50; and λ = 0.5. For this experiment, Movielens dataset had its contextual information (timestamp), decoded into hours, representing all hours from a day. The input matrices X_u, X_c, and X_a are rating matrix, contextual matrix, and content-feature matrix, respectively. Tables A.1, A.2, e A.3 show the performance of CHNMF and state-of-art for top-5 and top-10 recommendations.

[1] Due to memory limitation, it was not possible to reproduce the results on the larger datasets with the required setup

Algorithms	LDOS-CoMoDa			InCarMusic			Frappe			Movielens		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
UISplitting	0.0006	0.0030	0.0010	0.0035	0.0175	0.0058	0.0094	0.1560	0.0177	NA^2	NA^2	NA^2
DSPF	0.0138	0.0690	0.0230	0.1008	0.0504	0.0672	0.00261	0.0984	0.0412	NA^2	NA^2	NA^2
CAMF-C	0.0008	0.0043	0.0013	0.0045	0.0229	0.0075	0.1384	0.5582	0.2218	0.0003	0.0008	0.0004
CSLIM-ICS	0.0026	0.0131	0.0043	0.0031	0.0159	0.0051	NA^2	NA^2	NA^2	0.0943	0.0257	0.0403
CSLIM-LCS	0.0031	0.0157	0.0051	0.0049	0.0247	0.0081	NA^2	NA^2	NA^2	0.0018	0.0005	0.0009
CLSIM-MCS	0.0023	0.0117	0.0038	0.0028	0.0143	0.0046	NA^2	NA^2	NA^2	0.0017	0.0004	0.0006
LCE	0.1268	0.1368	0.1316	0.2111	0.1874	0.1985	0.5952	0.5749	0.5848	0.2000	0.1900	0.1948
CoNMF	0.1254	0.1467	0.1352	0.1943	0.1563	0.1732	0.5888	0.5735	0.5810	0.1988	0.1805	0.1892
MultiNMF	0.1305	0.1775	0.1504	0.1867	0.1743	0.1803	0.5830	0.5731	0.5780	0.1901	0.1800	0.1849
CHNMF	**0.1373**	**0.2033**	**0.1639**	**0.2222**	**0.1996**	**0.2103**	**0.5986**	**0.5763**	**0.5872**	**0.2032**	**0.1989**	**0.2010**

Table A.1: Top-5 Recommendations

Algorithms	LDOS-CoMoDa			InCarMusic			Frappe			Movielens		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
UISplitting	0.0011	0.0111	0.0020	0.0062	0.0628	0.0112	0.0004	0.0032	0.0007	NA^2	NA^2	NA^2
DSPF	0.0005	0.0055	0.0009	0.0070	0.0707	0.0127	0.0003	0.0023	0.0005	NA^2	NA^2	NA^2
CAMF-C	0.0009	0.0094	0.0016	0.0073	0.0735	0.0132	0.0006	0.0046	0.0010	0.0017	0.0008	0.0010
CSLIM-ICS	0.0024	0.0094	0.0038	0.0038	0.0380	0.0069	NA^2	NA^2	NA^2	0.0471	0.0257	0.0332
CSLIM-LCS	0.0025	0.0255	0.0045	0.0037	0.0373	0.0067	NA^2	NA^2	NA^2	0.0017	0.0009	0.0017
CLSIM-MCS	0.0024	0.0240	0.0043	0.0028	0.0287	0.0051	NA^2	NA^2	NA^2	0.0017	0.0009	0.0011
LCE	0.1389	0.1189	0.1281	0.1999	0.1190	0.1491	0.2997	0.1599	0.2085	0.2100	0.1974	0.2035
CoNMF	0.1188	0.1366	0.1270	0.1983	0.1189	0.1486	0.2992	0.1444	0.1947	0.2005	0.1901	0.1951
MultiNMF	0.1364	0.1183	0.1267	0.1970	0.1183	0.1478	0.2981	0.1451	0.1951	0.2009	0.1807	0.1902
CHNMF	**0.1399**	**0.1191**	**0.1286**	**0.2091**	**0.1194**	**0.1520**	**0.3020**	**0.1639**	**0.2124**	**0.2181**	**0.2000**	**0.2086**

Table A.2: Top-10 Recommendations

Algorithms	LDOS-CoMoDa	InCarMusic	Frappe	Movielens
UISplitting	0.0032	0.0295	0.0004	NA^2
DSPF	0.0050	0.0428	0.0012	NA^2
CAMF-C	0.0034	0.0232	0.5716	0.0008
CSLIM-ICS	0.0122	0.0181	NA^2	0.0034
CSLIM-LCS	0.0122	0.0254	NA^2	0.0107
CSLIM-MCS	0.0116	0.0134	NA^2	0.0011
LCE	0.3232	0.1013	0.8019	0.1119
CoNMF	0.3201	0.0947	0.6179	0.1001
MultiNMF	0.3227	0.0962	0.6111	0.1099
CHNMF	**0.3366**	**0.1080**	**0.8048**	**0.1221**

Table A.3: NDCG Performance

CHNMF has achieved a comparable performance as LCE, with a slight improvement, due to the combination of three matrices: rating, content features and contextual information. The context plays an important role in achieving better precision score, hence it shows in which conditions a target user *u* prefers to play a specific media. Furthermore, CoNMF and MultiNMF has shown approximate values of ranking quality and effectiveness, however under-performed CHNMF.

Pre-filtering and contextual modeling techniques presented poor performance, hence it does not incorporate content feature information. CSLIM method did not present significant results for Frappe dataset during the experiments, due to the broad range ratings. While, CAMF-C showed a good

NDCG value due the number of items combined with high contextual information. However, it did not overcome the result produced by the collective approaches.

Furthermore, Fig. A.1 presents the computational complexity analysis between ALS and CHNMF, comparing the convergence time (in logarithmic scale) against number of factors k. CHNMF had a better performance of 33% for Frappe and 34% for Movielens datasets compared to ALS. However, ALS had a better performance of 25% for LDOS-CoMoDa and 66% for InCarMusic datasets in comparison with CHNMF. Moreover, in both methods it was observed time increases linearly when compared with the number of factors. CHNMF performs better than ALS in larger datasets because Frappe and Movielens have denser matrices than LDOS-CoMoDa and InCarMusic.

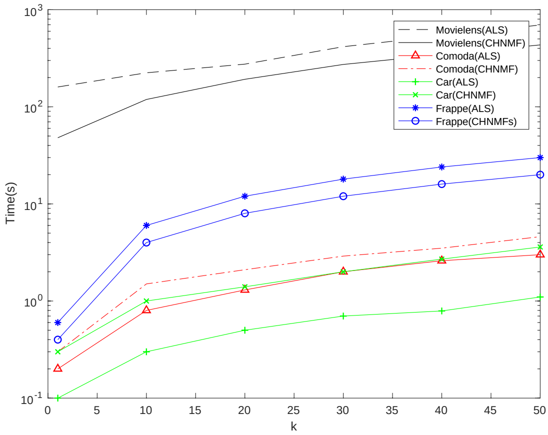


Fig. A.1: Convergence Time

8 Conclusions

We proposed CHNMF for a context-aware recommender system aggregating ratings, content features and contextual information in a common latent space. Furthermore, we introduced Barzilai-Borwein optimization into recommender systems combined with multiplicative update rules. Finally, we have experimentally shown the proposed methods, and generally outperform the state-of-the-art approaches considering the 4 datasets, LDOS-CoMoDa, InCarMusic, Frappe, and Movielens.

The top-N were addressed using three different matrices as input for CHNMF. We observed the content features, contexts, and ratings, when combined, play an important role for the user engagement: *users who rated an item* *i from an attribute a, and context c, tend to preferentially engage with each other about the same item in a specific context*.

We would like to extend CHNMF to offer explainable recommendations in natural language, presenting *why* the user receives a certain recommendation. Furthermore, optimizing the learning model may benefit CHNMF to perform better in scalable systems.

9 Acknowledgments

The authors wish to acknowledge the financial support and the fellow scholarship given to this research from the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0)

References

[1] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recc Sys. handbook*. Springer, 2011, pp. 217–253.

[2] Y. Zheng, R. Burke, and B. Mobasher, "Splitting approaches for context-aware recommendation: An empirical study," in *Proc. of the 29th Annual ACM Symp.on Applied Computing*, ser. SAC '14. New York, NY, USA: ACM, 2014, pp. 274–279. [Online]. Available: http://doi.acm.org/10.1145/2554850.2554989

[3] V. Codina, F. Ricci, and L. Ceccaroni, "Distributional semantic pre-filtering in context-aware recommender systems," *User Modeling and User-Adapted Interaction*, vol. 26, no. 1, pp. 1–32, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1007/s11257-015-9158-2

[4] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, "Incarmusic: Context-aware music recommendations in a car," in *E-Commerce and Web Technologies*. Springer, 2011, pp. 89–100.

[5] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proc. of Intl. Conf. on Data Mining*. SIAM, 2013, pp. 252–260.

[6] X. He, M.-Y. Kan, P. Xie, and X. Chen, "Comment-based multi-view clustering of web 2.0 items," in *Proceedings of the 23th International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: ACM, 2014, pp. 771–782. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567975

[7] Y. Huang, H. Liu, and S. Zhou, "An efficient monotone projected barzilai–borwein method for nonnegative matrix factorization," *Applied Mathematics Letters*, vol. 45, pp. 12–17, 2015.

[8] Y. Zheng, B. Mobasher, and R. Burke, "Similarity-based context-aware recommendation," in *Intl. Conf. on Web Inf. Systems Eng.* Springer, 2015, pp. 431–447.

[9] L. R. Drumond, E. Diaz-Aviles, L. Schmidt-Thieme, and W. Nejdl, "Optimizing multi-relational factorization models for multiple target relations," in *Proceedings of the 23th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '14. New York, NY, USA: ACM, 2014, pp. 191–200. [Online]. Available: http://doi.acm.org/10.1145/2661829.2662052

[10] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: ACM, 2014, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/2645710.2645751

[11] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA journal of numerical analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[12] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic, "Database for contextual personalization," *Elektrotehniški vestnik*, vol. 78, no. 5, pp. 270–274, 2011.

[13] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, "Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild," *CoRR*, vol. abs/1505.03014, 2015.

[14] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 19:1–19:19, Dec. 2015. [Online]. Available: http://doi.acm.org/10.1145/2827872

Paper B

Predicting Visitors Using Location-Based Social Networks

Muhammad Aamir Saleem, Felipe Soares Da Costa, Peter Dolog, Panagiotis Karras, Torben Bach Pedersen, Toon Calders

The paper has been published in the
Proceedings of 19th IEEE International Conference on Mobile Data Management (MDM'18), pp. 245–250, 2018.

Abstract

Location-based social networks (LBSN) are social networks complemented with users' location data, such as geo-tagged activity data. Predicting such activities finds application in marketing, recommendation systems, and logistics management. In this paper, we exploit LBSN data to predict future visitors at given locations. We fetch the travel history of visitors by their check-ins in LBSNs and identify five features that significantly drive the mobility of a visitor towards a location: (i) historic visits, (ii) location category, (iii) time, (iv) distance, and (v) friends' activities. We provide a visitor prediction model, CMViP, based on collective matrix factorization and influence propagation. CMViP first utilizes collective matrix factorization to map the first four features to a common latent space to find visitors having a significant potential to visit a given location. Then, it utilizes an influence-mining approach to further incorporate friends of those visitors, who are influenced by the visitors' activities and likely to follow them. Our experiments on two real-world data-sets show that our methods outperform the state of art in terms of precision and accuracy.

User	Location	Time
u_1	l_1	13:25 11/12/2017
u_1	l_1	14:30 12/12/2017
u_1	l_3	13:05 14/12/2017
u_2	l_1	13:10 11/12/2017
u_2	l_1	15:10 14/12/2017
u_3	l_1	14:20 11/12/2017
u_3	l_2	13:16 12/11/2017
u_3	l_3	16:20 14/12/2017
u_4	l_2	15:15 11/12/2017

Location	Category	Coordinates
l_1	Sports	42.99,-71.46
l_2	Arts	42.98,-71.45
l_3	Sports	42.97,-71.44

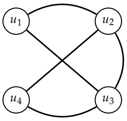


Fig. B.1: Toy example: Checkins (left), Location categories (top right) and social graph (bottom right)

1 Introduction

Social network analysis allows the provision of diverse recommendations to users, e.g., friends and activities. At the same time, the pervasiveness of location-aware devices allows users of online social networks to share geo-tagged contents such as their current location. Such online social networks that exploit location information are called location-based social networks (LBSNs). The geo-tagging of activities in LBSNs provides an opportunity to capture and utilize the mobility behaviour of users such as to provide *location-based recommendations* [1]. For instance, one can analyze the historical activities of users and recommend locations to them for their potential visit; in such cases, the focus is on providing recommendations to the users. An overlooked perspective to the recommendation is the *prediction* of users that will visit a given location; such a perspective finds application in event planning, traffic management, and mobile phone capacity planning. For instance, consider the following example:

> **Example 1.1**
> A cinema owner wants to know the persons who would choose to watch the movie *Pirates of Caribbean* at a cinema at a certain time.

To answer such queries, we need to predict that whether a user will visit a location of a particular category, e.g., cinema, in a particular region and at a particular time. An illustration of this example is shown in Fig. B.1. Here, we show the users' check-ins in the form of triplets: user Id, location Id, and check-in time. We also show the characteristics of locations, i.e., location id, category, and GPS coordinates and a social graph depicting friends of users in LBSNs. Here, in order to predict the visitors at l_1 at t_1 (hour of the day):

13:00, we compute the potential of users based on their check-ins at l_1, at locations of category *Sports* (*l_1's category*) and at time t_1. Based on these values, u_1 and u_2 are considered as potential visitors of l_1 at t_1 as they have the highest number of check-ins under these conditions. In this paper, we address such a problem: *Given a location, its category, and a time period, predict the visitors to this location within the given time period using LBSN data on past mobility.*

To address this problem, we need to identify which features in past mobility data affects users' mobility. To that end, we analyze the available LBSN data and identify the following mobility-affecting features: (i) historic visits, (ii) location category, (iii) time, (iv) distance, and (v) friends' activities.

We provide a novel model CMViP for predicting visitors that combines collective non-negative matrix factorization approach with an influence diffusion model. Initially, we compute a set of frequency matrices, capturing the frequency of *users'* visits at the given location(s), at locations of similar *categories*, as well as visiting *times* and *distances* among users' current locations and the location(s) for which visits are to be predicted. Then, we decompose these matrices into four non-negative low-rank matrices, i.e, H_U, H_C, H_T, and H_D, respectively, and a common latent space matrix W, so that $W \times H_U$, $W \times H_C$, $W \times H_T$, and $W \times H_D$ give approximated scores according to user's preferences. We use a linear combination of these matrices to form a score matrix WH_X that represents the potential of each user to visit the given location ("visit score"). We consider all users having significant visit scores and find their friends that are presumably influenced by their activities and likely to follow them in the given time period. To do so, we compute the *influence probabilities* of these users on their friends in the LBSN by a Bernoulli-distribution based partial credit distribution and time constraint model. Users having a significant visit score, along with their significantly influenced friends, are considered as potential visitors to the location. It is worth noting that a user u may not influence another user v alone, yet, taken together with another user x, they may influence v; that is the reason why we first find all the users having a significant visit score and then fetch their influenced visitors.

In summary, we make the following contributions.

- We propose a visitor prediction model based on collective non-negative matrix factorization and influence propagation (CMViP).
- We provide an extensive experimental evaluation of our proposed methods on real-world data-sets to showcase their precision and accuracy.

The rest of the paper is organized as follows. Section 2 covers related work in the domain. Section 3 provides preliminaries and give details on CMViP. We provide details on our solution framework in Section 4, present evaluation results in Section 5, and conclude in Section 6.

2 Related Works

In this section, we survey the existing studies on prediction of visitors.

Lasek et al. [2] focus on predicting visitors to locations with applications such as estimating restaurant sales and demands. Sellers and Shmueli [3] propose Poisson regression models [4, 5] to predict the number of customers at a restaurant during a certain time period. Similarly, Morgan and Chintagunta [6] apply a regression model to predict the number of visitors throughout a calendar year. Koshiba et al. [7] use a Bayesian network to predict users at a location. For prediction, they exploit features such as categories of locations, historic visits, and demographic information of visitors. Authors in [8–10] further incorporates sequential transitions of users from one location to another for prediction of visitors. To do so they exploit factorized personalized Markov chain models (FPMC) and metric embedding to model users' preferences and POI transitions.

Most recently, Feng et al. [11] proposed an embedding model, POI2Vec, that predicts visitors to a Point of Interest (POI). POI2Vec modifies the Word2vec technique for word embeddings to learn POI representations by considering their geographical influence, user preferences, and sequential transitions, using the GPS coordinates of locations and visiting timestamps. POI2Vec is the most relevant previous work to ours, as it also aims to predict visitors at a location using LBSNs. However, our approach is technically different. While POI2Vec is based on a vector embedding of locations, we employ a collective matrix factorization [12, 13] to find joint embeddings of users' historic visits, categories, time, and distance features. Furthermore, we combine the prediction outcome of the factorized model with an influence propagation method to fetch friends of users that are likely to visit a certain location as well. Our comparison in Section 5.5 shows that our proposed model CMViP outperforms all variants of POI2Vec in both accuracy and precision.

3 Problem Formulation

Here, we define preliminaries, the CMViP model, and formulate our problem.

3.1 Preliminaries

Definition 3.1. A **point of interest** is a geographical location (e.g., an amenity) represented by a quadruple $(l, lat, long, C)$, where l is the identifier, lat and $long$ are the latitude and longitude of the GPS coordinates of the center of the POI, and C is a set of categories that this location belongs to (e.g., "Food", "Restaurant", and "French cuisine". Such categories are usually assigned by the visitors, and may thus change dynamically. However, we consider each location to have a fixed set of categories. A set of POIs is denoted as L.

Definition 3.2. A **user** is a person that visits POIs. A set of users is denoted by U.

Definition 3.3. An **activity** refers to a visit or check-in of a user $u \in U$ at a location l at a discretized time interval t, represented as a triplet (u, l, t). The set of all activities over U and L is denoted $A(U, L)$.

Definition 3.4. An **Location-Based Social Network (LBSN)** over U and L consists of a graph $G_S(U, F)$, called the *social graph*, where $F \subseteq \{\{u, v\} | u, v \in U\}$ represents friendships between users, and set of activities $A(U, L)$. It is denoted as $LBSN(G_S, A)$.

3.2 CMViP

We utilize five features to predict a visit of a user u to a location l: (i) historic visit frequency of u at l, (ii) historic visit frequency of u at locations having similar categories as l, (iii) visit frequency of u at time of day T (iv) distance of l from current location of u, and (v) influence of u on friends. Next, we provide the scores for each of these features.

Visit frequency score

People tend to visit locations of their interests [14]. We evaluate the interest of a user u in a location l based on a *visit frequency score*, given by:

$$Y_U(u, l) = \frac{|A(u, l)|}{|A(u)|} \tag{B.1}$$

where $A(u, l)$ is the set of activities of u at l and $A(u)$ is all activities of u.

> **Example 3.1**
> Consider the toy example, given in Fig. B.1. Here, $Y_U(u_1, l_1) = 2/3, Y_U(u_2, l_1) = 2/2, Y_U(u_3, l_1) = 1/3$ and $Y_U(u_4, l_1) = 0$

Location category score

People like to visit locations of their general interest, identified by a location's category, such as "museum" and "Chinese restaurant". To incorporate this effect we compute the *location category score*, given by the following equation.

$$Y_C(u,c) = \frac{\sum_{l|c \in l.C} |A(u,l)|}{|A(u)|} \tag{B.2}$$

where $l.C$ is the set of categories of l.

Example 3.2
Consider the running example, given in Fig. B.1. Let $c_1 = \mathit{Sports}$. Here, $Y_C(u_1, c_1) = 3/3, Y_C(u_2, c_1) = 2/2, Y_C(u_3, c_1) = 2/3$, and $Y_C(u_4, c_1) = 0$.

Visit time score

Another feature that drives the visits of users is time. Usually, users visit a location at similar times of the day [14]. We compute the potential of a user to visit a location at a given time (considering time in a granularity of hours) by a *visit time score*, as follows.

$$Y_T(u,T) = \frac{|\{(u,l,t') \in A | t'.hour = T\}|}{|A(u)|} \tag{B.3}$$

where $t'.hour$ and T represent check-in time and given time in hours, respectively.

Example 3.3
Consider the running example, given in Fig. B.1. We take hours of visit time for computing Y_t. Let's $T_1 = 1300$, then $Y_T(u_1, T_1) = 2/3, Y_T(u_2, T_1) = 1/2, Y_T(u_3, T_1) = 1/3$ and $Y_T(u_4, T_1) = 0$.

Distance score

People tend to visit nearby places [14]. To capture this effect, we measure the distance between the current location of the user and the location which we aim to evaluate. We utilize the GPS coordinates of locations to measure distances and compute a *distance score* by the following equation.

$$Y_D(u,D) = 1 - \frac{D(l_c, l)}{D_{max}} \tag{B.4}$$

where l_c represents the current location of user u, l shows the location for which the visit is to be predicted, $D(l_c, l)$ shows the distance between l_c and l, and D_{max} is the maximum distance u traveled to visit any location.

Example 3.4
Consider the running example, given in Fig. B.1. Let the location for which we aim to predict visitors is l_1, then $Y_D(u_1, l_1) = 1 - \frac{1}{1} = 0$. Similarly, $Y_D(u_2, l_1) = 0, Y_D(u_3, l_1) = 0.5$ and $Y_D(u_4, l_1) = -\infty$.

Next, we combine the aforementioned scores using a linear combination to compute the overall potential of users. We call this potential a *visit score* denoted by Y_S. It is given by:

$$\begin{aligned} Y_S(u_1, l_1, c_1, T_1) = \alpha.Y_U(u_1, l_1) + \beta.Y_C(u_1, c_1) + \\ \gamma.Y_T(u_1, T_1) + \eta.Y_D(u_1, l_1) \end{aligned} \tag{B.5}$$

Here, α, β, γ, and η are the coefficients for visit frequency, location category, visit time, and distance respectively, showing their corresponding importance in predicting visitors. $\{\alpha, \beta, \gamma, \eta\} \in [0, 1]$, however, we provide optimal values for these parameters in Section 5. We utilize Equation B.5 to compute the visit score for all the users in LBSN. Then, we prune the users with less potential of visiting the location based on their visit score. To do that we use a threshold θ. All the users having visit score greater than θ are considered potential visitors U_P and given by:

$$U_P(l, c, T) = \{u | Y_S(u, l, c, T) \geq \theta\} \tag{B.6}$$

where θ is a threshold for visit score.

Example 3.5
Consider the toy example, given in Fig. B.1. Let $\alpha = 0.25, \beta = 0.25, \gamma = 0.25, \eta = 0.8$, and $\theta = 0.8$. Then, $Y_S(u_1, l_1, c_1, t_1) = 0.84, Y_S(u_2, l_1, c_1, t_1) = 0.875, Y_S(u_3, l_1, c_1, t_1) = 0.66$ and $Y_S(u_4, l_1, c_1, t_1) = -\infty$.

Friends Influence

Users tend to follow the activities of their friends [1]. To capture this effect, we compute the influence of users on their friends, i.e., assess their potential to persuade friends to follow their activities.

We consider a user v to be influenced by his friend u if u visits a location l and v visits the same location after u within a particular time. To find such influence, we compute influence probabilities using Bernoulli distribution based on partial credit distribution and discrete time constraint models [15]. According to this model, the influence probability is measured as the ratio of the number of successful attempts to persuade the influenced user to follow the influencer user's activities over the total number of trials. Considering that a user can be influenced by multiple sources for an activity, the influence credit for each followed activity is distributed among all influencer users. Further, as the influence probability depends on time, we incorporate a discrete time constraint model which ensures that a user can influence other users only within a given time window. We consider the time window $\omega = 1$, as we capture the visit times in hours. The influenced friends of a user u are:

$$I(u) = \{v | p(u,v) \geq \xi \wedge (u,v) \in F\} \tag{B.7}$$

where $p(u,v)$ is the influence probability of u on v, ξ is a threshold representing minimum influence to persuade a user to follow an activity, and F is the set of friends pairs in an LBSN. A user v may not be influenced by a single user u, but by many other users taken together. Thus, we compute the influence of all potential visitors on all of their friends together, using the following equation.

$$I(U_P) = \{v | \sum_{u \in U_P} Y(u,v) \geq \xi \wedge (u,v) \in F\} \tag{B.8}$$

where U_P is the set of users having a significant visit score. Thus, $I(U_P)$ are also considered as potential visitors. So, the total potential visitors are given by the union of U_P and $I(U_P)$ It is given by: $U_P \cup I(U_P)$.

> **Example 3.6**
> Consider the toy example, given in Fig. B.1. Let $U_P = \{u_1, u_2\}, \omega = 1h$ and $\xi = 0.2$. Then, $I(U_P) = \{u_3\}$ as $p(U_P, u_3) = 0.3 \geq \xi$ Because, u_3 follows u_1 and u_2 for visiting l_1 and follows u_1 for visiting l_3 within ω. So, the predicted visitors are $\{u_1, u_2, u_3\}$.

3.3 Problem Statement

Next, we formally define our problem statement as follows:

Problem 3.1. *Given a LBSN(G_S, A), a location l, the categories C of l, a time interval T, and the distance between the current location of the user and the next location D, predict the users that will visit l within T.*

Once we obtain potential visitors, we can utilize it for several other use-cases such as finding the number of visitors at a location or finding the types of visitors etc.

4 Solution Framework

In this section, we provide details on CMViP and cover the techniques and steps it uses for prediction of visitors.

4.1 Non-Negative Collective Matrix Factorization

The LBSN data is often sparse (shown in Table B.1). This badly affects the prediction accuracy because of limited check-in data availability. To avoid this, we use matrix factorization which helps to find hidden latent representations in a common space and thus, overcomes the sparsity issue. Since we consider multiple features: visit frequency, categories, time and distance to consider for prediction thus, we need to collectively factorize them. To do that, we use collective matrix factorization. Furthermore, because we do not anticipate any negative values in our data, so, we use non-negative collective matrix factorization. To do so, we decompose each of the four frequency matrices mentioned above into a common latent space matrix: W and corresponding feature latent space matrix H. Each row of W represents the relation between a visitor and the number of latent factors. Similarly, each column of H represents the relation between the number of latent factors and the features.

Next, we formally define the non-negative collective matrix factorization for the four frequency matrices:

$$\begin{aligned} min : f(W) = \frac{1}{2}[\alpha\|Y_U - WH_U\|_2^2 + \beta\|Y_C - WH_C\|_2^2 \\ +\gamma\|Y_T - WH_T\|_2^2 + \eta\|Y_D - WH_D\|_2^2 \\ +\lambda(\|W\|_2 + \|H_U\|_2 + \|H_C\|_2 + \|H_T\|_2 + \|H_D\|_2)] \\ s.t. W \geq 0, H_U \geq 0, H_C \geq 0, H_T \geq 0, H_D \geq 0 \end{aligned} \tag{B.9}$$

where W represents the common latent space during the decomposition of Y_U, Y_C, Y_T, and Y_D , while $\{\alpha, \beta, \gamma, \eta\} \in [0, 1]$ are hyper-parameters that control the importance of each matrix during the factorization. Setting them as 0.25 gives equal importance to decomposition matrices, while different values give more importance to the factors having more values. The remaining terms are Tikhonov regularization [16] of W,H_U, H_C, H_T, and H_D controlled by the hyper-parameter $\lambda \geq 0$. It is used to enforce the smoothness of the solution and avoid over-fitting. Note, the matrices H_U, H_C, H_T, and H_D when multiplied by W give us an approximated score of the input frequency matrices.

Multiplicative Update Rules

The proposed model applies multiplicative update rules as regularization term of H_U, H_C, H_T, H_D, and W. This technique updates the scores in each iteration until reaches the stationary point. Formalizing the update rules, we have:

$$\begin{aligned}
W = \frac{[\alpha Y_U H_U^T + \beta Y_C H_C^T + \gamma Y_T H_T^T + \eta Y_D H_D^T]}{[\alpha H_U H_U^T + \beta H_C H_C^T + \gamma H_T H_T^T + \eta H_D H_D^T + \lambda I_k]} \\
H_U = (\alpha W^T W H_U + \lambda I_k)^{-1} \odot \alpha W^T Y_U \\
H_C = (\beta W^T W H_C + \lambda I_k)^{-1} \odot \beta W^T Y_C \\
H_T = (\gamma W^T W H_T + \lambda I_k)^{-1} \odot \gamma W^T Y_T \\
H_D = (\eta W^T W H_D + \lambda I_k)^{-1} \odot \eta W^T Y_D
\end{aligned} \tag{B.10}$$

where $\div$ and $\odot$ corresponds to the left division and element-wise matrix product, respectively, and I_k is the identity matrix with $k \times k$ dimensions.

Each iteration of the proposed model gives us a solution for the pair-wise division. As we map any negative values to zero, the W matrix becomes non-negative after each update. Furthermore, the objective function and the delta decrease on each iteration of the above update rules, guaranteeing the convergence into a stationary point.

4.2 Prediction of Visitors

The preferences scores for a user u is given by $s_i = WH_x$, where W are the factors in the common latent space that explain the preferable places of $u_i \in U$, and H_x represents the relation among the output matrices: H_U, H_C, H_T, and H_D. It is given by the following equation. t is worth noting that the following equation (Equation B.11) provides an approximate version of Equation B.5.

$$\begin{aligned}
WH_X = \alpha.WH_U(u,l) + \beta.WH_C(u,l) \\
+\gamma.WH_T(u,l) + \eta.WH_D(u,l),
\end{aligned} \tag{B.11}$$

where, $+$ corresponds to the element-wise sum, while $\{\alpha, \beta, \gamma, \eta\} \in [0,1]$ are hyper-parameters controlling the importance of each factorized matrix. The sum of hyper-parameters are set to be 1.

Next, we prune all the users having visit score less than the threshold θ. Users having visit score more than θ are considered potential visitors U_P given in equation B.6. We further utilize them to find their influenced visitors using the algorithm given in Section 3.2. We combine the potential visitors with their influenced visitors and predict them as visitors of the location.

5 Experiments

In this section, we provide a detailed experimental evaluation of the solutions using two real-life data-sets. We first describe the data-sets, then we present the data prepossessing and preparation. Finally, we provide results for our experimental evaluation.

5.1 Data-sets

We utilize two real-world data-sets: 1) Foursquare data-set which is smaller and denser and 2) Weeplaces data-set [17] which is larger and sparser. Table B.1 shows the statistics of the data. Each of these data-sets contains social friendship graph and an ordered list of check-ins of users at locations. We further cluster locations to find POIs [18].

	Users	Locations	Checkins	POIs	Friend pairs	Duration
FourSquare	4K	0.2M	0.47M	0.12M	32K	1322 days
Wee	16K	0.9M	8M	0.76M	0.1M	2796 days

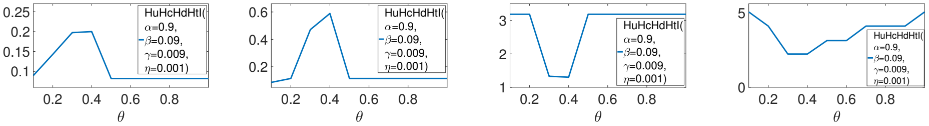
Table B.1: Data-set Statistics

5.2 Evaluation Measures

We utilized the cross-validation approach to evaluate CMViP using four measures: precision, recall, MAE and RMSE. We divided the data-sets based on time stamp of check-ins such that each part consists of check-ins of one month. We trained CMViP on one month and test its performance on the data-set of the next month in sequence. The process was repeated for all the parts and average results are reported. To compute precision and recall, we compute confusion matrix based on following possible outcomes for users for the input parameters: correctly predicted visitors as true positives (TP), correctly predicted non-visitors as true negatives (TN), wrongly predicted visitors as false positives (FP), and wrongly predicted non-visitors as false negatives (FN). Likewise, to calculate the error applying MAE and RMSE, we use the number of visitors against queries in the test data-set in comparison to the training data-set.

5.3 Parameter Analysis

CMViP uses several hyper-parameters while computing the visit score and incorporating the influence of potential visitors as shown in Equations B.5, B.6, and B.7. In this section, we provide optimal values of these parameters as shown in Section 4.



(a) Precision **(b)** Recall **(c)** MAE **(d)** RMSE

Fig. B.2: Threshold Analysis for Wee data-set

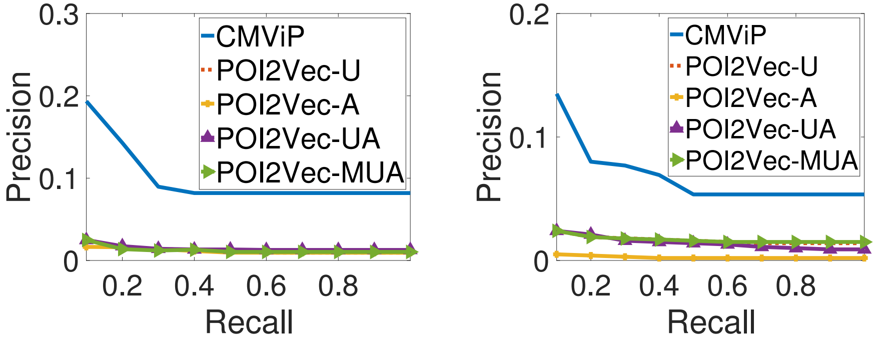
We performed iterative empirical experiments to set the optimal hyper-parameters values, where we define: $k = 50$, $\alpha = 0.9$, $\beta = 0.09$, $\gamma = 0.009$, $\eta = 0.001$, and $\lambda = 0.5$. Small values of k under-fit, on the other hand, large values of k over-fit the data and lead to poor performance. The visit score has higher hyper-parameter because it has shown to increase the accuracy for predicting the visitors. Furthermore, λ with lower value keeps the accuracy stable.

Predicting the number of potential visitors lead us to define $\theta = 0.4$, which presents better accuracy score as shown in Figure B.2 for Wee data-set. Similar results are exhibited for Foursquare data-set, hence we omit them due to space constraints. The accuracy decreases by increasing or decreasing the value of θ beyond the optimal value. The reason behind this is that the model ignores actual visitors when θ is higher and consider users with low potential of visiting the location when θ is lower. Furthermore, we find the optimal value of influence probability threshold. We assume that users influenced with probability greater than this threshold will follow their influential friends' activities. We consider the value of this threshold 0.04 as we assume that the top 10% the most influenced visitors follow the activities. We use these values of the parameters for evaluating CMViP.

5.4 Competitors

We compare the performance of CMViP against following four different variants of POI2Vec using corresponding optimal values for each variant.

- PI2Vec-**U**: utilizes user's preferences.
- PI2Vec-**A**: considers only users with recent locations.
- PI2Vec-**UA**: incorporates both preference and recent locations.
- PI2Vec-**MUA**: applies aggregation to incorporate preferences and transition among locations.



(a) Wee

(b) Foursquare

Fig. B.3: Precision-Recall Curve

5.5 Results

Figure B.3 presents the experimental results considering precision-recall curve, where we observed, CMViP outperforms all variants of **POI2Vec**: (1) **U**, (2) **A**, (3) **UA**, and (4) **MUA**. CMViP outperforms **U** to predict the potential visitors. Even though **U** uses user's preferences to predict visitors, it does not consider other features as categories, time, and distance. This means CMViP has better user modeling in the vectorial space. **U** first learns the location representation, and the user representation is learned when the location representation is fixed. However, as CMViP presented better performance than **U**, it shows the collective factorization and linear model is more reasonable for this task. **A** considers only recent check-ins presenting a poor performance in comparison with other techniques since old check-ins present an important role to determine the future visitors. The results show it is important to consider both users with recent and old check-ins. **MUA** outperforms **UA**, indicating that combining user preference and sequential transition plays an important role to predict potential visitors. However, it under-performs CMViP, because CMViP uses categories, time, and distance to jointly predict the visitors. Furthermore, CMViP incorporates visitors' influenced friends as potential visitors which further improves the accuracy.

Observing the Figure B.3, we can conclude that CMViP significantly improves the prediction accuracy by collectively learning the location and visitor's latent factors and incorporating visitor's influenced friends using the linear model. Considering that LBSN data is often sparse, as shown in Table B.1, it is crucial for a model to perform better on sparse data. CMViP consistently improves the visitors' prediction for locations with very few visitors as observed on sparse data-set: Wee. This happens, because CMViP explicitly makes use of jointly process latent factors to learn better latent representations, especially when visitor-location history data is sparse.

6 Conclusions

We proposed a model, CMViP, that predicts visitors given a location, its category, and visit time. CMViP employs non-negative collective matrix factorization to find the potential of visitors and further leverages these potential visitors to find their influence on their friends in social networks and thereby incorporate highly influenced visitors as potential visitors.

We have empirically shown that the CMViP outperforms the state-of-the-art approaches using two real-world data-sets. First, we evaluate the performance of CMViP for predicting visitors using precision and recall measures and then evaluate the accuracy of predicting the number of visitors using RMSE and MAE. Our results show that CMViP outperforms state-of-the-art methods in precision and recall up to 10 times. We further show the significance of considered features for visitor prediction.

In the future, we plan to extend the proposed model considering more contextual features, such as week-days and weekends. Further, we plan to improve performance by pruning users with less potential visitors.

7 Acknowledgments

This research has been funded in part by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College"(IT4BI-DC). Felipe Soares da Costa is supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0).

References

[1] M. A. Saleem, R. Kumar, T. Calders, X. Xie, and T. B. Pedersen, "Location influence in location-based social networks," in *WSDM*, 2017, pp. 621–630.

[2] A. Lasek, N. Cercone, and J. Saunders, "Restaurant sales and customer demand forecasting: Literature survey and categorization of methods," in *Smart City 360°*, 2016.

[3] K. F. Sellers and G. Shmueli, "Predicting censored count data with compoisson regression," *Robert H. Smith School Research Paper N°. RHS-06-129*, 2010.

[4] S. Coxe, S. G. West, and L. S. Aiken, "The analysis of count data: A gentle introduction to poisson regression and its alternatives," *Journal of personality assessment*, 2009.

[5] J. Wulu, K. Singh, F. Famoye, T. Thomas, and G. McGwin, "Regression analysis of count data," *Indian Society of Agricultural Statistics*, 2002.

[6] M. S. Morgan and P. K. Chintagunta, "Forecasting restaurant sales using self-selectivity models," *Journal of Retailing and Consumer Services*, 1997.

[7] H. Koshiba, T. Takenaka, and Y. Motomura, "A service demand forecasting method using a customer classification model," *The Philosopher's Stone for Sustainability*, 2013.

[8] J.-D. Zhang, C.-Y. Chow, and Y. Li, "Lore: Exploiting sequential influence for location recommendations," in *SIGSPATIAL*, 2014.

[9] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation." in *IJCAI*, 2013.

[10] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new poi recommendation," in *IJCAI*, 2015.

[11] S. Feng, G. Cong, B. An, and Y. M. Chee, "Poi2vec: Geographical latent representation for predicting future visitors." in *AAAI*, 2017.

[12] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: ACM, 2014, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/2645710.2645751

[13] F. Costa and P. Dolog, "Hybrid learning model with barzilai-borwein optimization for context-aware recommendations," in *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida USA., May 21-23 2018.*, 2018, pp. 456–461.

[14] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *KDD*, 2011.

[15] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, 2010.

[16] A. N. Tikhonov, "Solution of incorrectly formulated problems and the regularization method," *Soviet Math. Dokl.*, vol. 4, pp. 1035–1038, 1963.

[17] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *CIKM*, 2014, pp. 739–748.

[18] M. A. Saleem, R. Kumar, T. Calders, X. Xie, and T. B. Pedersen, "Imaxer: A unified system for evaluating influence maximization in location-based social networks," in *CIKM*, 2017, pp. 2523–2526.

Paper C

Automatic Generation of Natural Language Explanations

Felipe Costa, Sixun Ouyang, Peter Dolog, Aonghus Lawlor

The poster has been published in the
Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion (IUI'18), pp. 57:1–57:2, 2018.

Abstract

An interesting challenge for explainable recommender systems is to provide successful interpretation of recommendations using structured sentences. It is well known that user-generated reviews, have strong influence on the users' decision. Recent techniques exploit user reviews to generate natural language explanations. In this paper, we propose a character-level attention-enhanced long short-term memory model to generate natural language explanations. We empirically evaluated this network using two real-world review datasets. The generated text present readable and similar to a real user's writing, due to the ability of reproducing negation, misspellings, and domain-specific vocabulary.

1 Introduction

A recommender system should provide accurate and relevant recommendations, but a good recommendation must be supported by interpretation. The explanation is the key factor to gain the trust of the user. An interpretable system has significant influence on a user's decision [1], and users tend to trust the opinion of others, especially when they describe personal experiences [2].

Current explainable recommendations propose to mine user's reviews to generate explanations. Nonetheless, they lack generating natural language expressions, hence the sentences are produced in a modular way. We aim to generate natural language explanations from reviews, aligning explanations and textual features such as aspects and sentiments, which influence the recommendation of different items. We exploit deep neural networks at character-level to generate explanations. These networks have recently shown good performance to generate sentences as presented by Karpathy *et. al.*, who use a variant of LSTM cells to generate text [3]. Karpathy *et. al.* presented encoding rating vectors of reviews in the training phase, allowing the system to calculate the probability of the next character based on the given rating. Later, Dong *et. al.* presented an efficient method to generate the next word in a sequence when it is added an attention mechanism, improving the performance for long textual sequences [4].

In this paper, we propose a character-level attention-enhanced long short-term memory (LSTM) model to generate personalized natural language explanations based on user-generated reviews. The model is trained using two real-world datasets: BeerAdvocate [5] and Amazon book reviews [4]. The datasets present user reviews describing their opinion about items in natural language. The explanations are adaptively composed by an encoder-side context vector, because our model learns soft alignments between generated characters and user-item relations, for example, ratings from a user to an item.

2 Interpretation model

The character-level explanation model presents (1) three modules: LSTM network, attention layer, and generator module; and (2) two input sources: review text and concatenated word embeddings of user, item, and rating, as presented in Fig. C.1. First, users and items embeddings are learned from *doc2vec* model, where characters of reviews are encoded as one-hot vectors, corresponding to the input time-steps of LSTM network. Second, the embeddings are concatenated with the outputs of LSTM, which are inputs for the following attention layer. Finally, the generator module produce sentences as explanations using outputs from the attention layer.

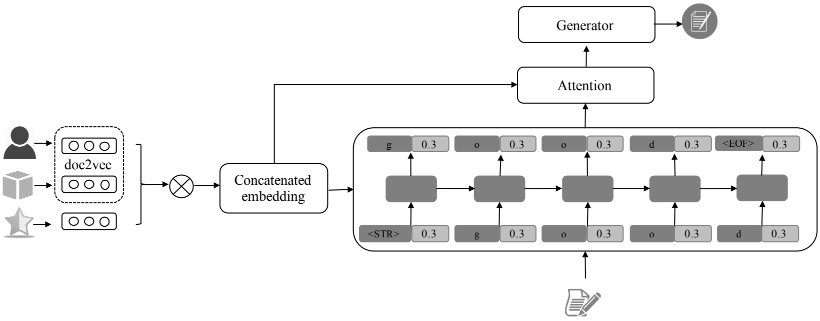


Fig. C.1: Personalized Explanation Generation Model Architecture

- **LSTM network** LSTM is an enhanced recurrent neural network (RNN) where information is transmitted from a neuron to the next neuron, and the corresponding neuron in the next layer simultaneously, as presented in Fig. C.1. LSTM was introduced to solve the long-term dependency problem, which causes vanishing gradient in conventional RNN [3].

- **Attention mechanism** The attention mechanism, adaptively learns soft alignments c_t between character dependencies H_t and attention inputs a. Equation C.1 formally defines the new character dependencies using attention layer $H_t^{attention}$ [4].

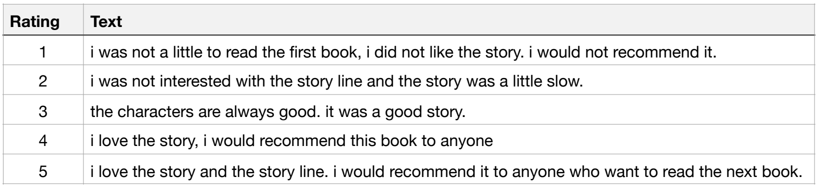
$$\begin{aligned} c_t &= \sum_i^a \frac{\exp(\tanh(W_s \odot [H_t, a_i]))}{\sum \exp(\tanh(W_s \odot [H_t, a_i]))} a_i \\ H_t^{attention} &= \tanh(W_1 \odot c_t + W_2 \odot H_t) \end{aligned} \tag{C.1}$$

- **Generating Text** The explanation is generated character by character. The characters are given by maximizing the *softmax* conditional probability p, based on the new character dependencies $H_t^{attention}$ [4], as presented in Equation C.2

$$p = \text{softmax}(H_t^{attention} \odot W + b), \quad char = \arg\max p \tag{C.2}$$

3 Results

The model was evaluated using two real-world datasets: BeerAdvocate and Amazon book reviews. The first experiment presents generated explanations

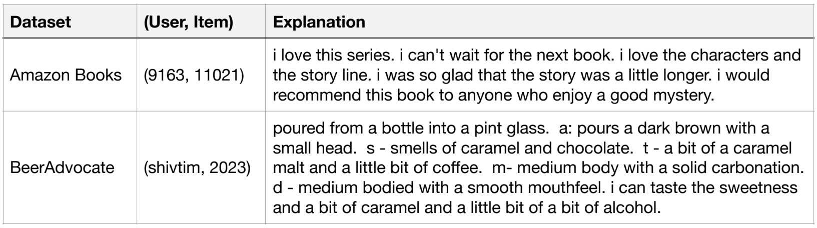


Rating	Text
1	i was not a little to read the first book, i did not like the story. i would not recommend it.
2	i was not interested with the story line and the story was a little slow.
3	the characters are always good. it was a good story.
4	i love the story, i would recommend this book to anyone
5	i love the story and the story line. i would recommend it to anyone who want to read the next book.

Fig. C.2: Rating Text Samples, from poorly rated (1) to highly rated (5).

given by the rating as attention mechanism to generate explanations with different sentiments, as presented in Fig. C.2.

The second experiment generates explanations for particular user-item pairs presented in Fig. C.3, where the user opinion about an item is generated in natural language. Finally, evaluating the generated explanations based on readability metrics in Fig. C.4. The readability metrics [6] measure how understandable the generated text is, where lower values correspond to an easy and understandable text.



Dataset	(User, Item)	Explanation
Amazon Books	(9163, 11021)	i love this series. i can't wait for the next book. i love the characters and the story line. i was so glad that the story was a little longer. i would recommend this book to anyone who enjoy a good mystery.
BeerAdvocate	(shivtim, 2023)	poured from a bottle into a pint glass. a: pours a dark brown with a small head. s - smells of caramel and chocolate. t - a bit of a caramel malt and a little bit of coffee. m- medium body with a solid carbonation. d - medium bodied with a smooth mouthfeel. i can taste the sweetness and a bit of caramel and a little bit of a bit of alcohol.

Fig. C.3: Generated Text Samples

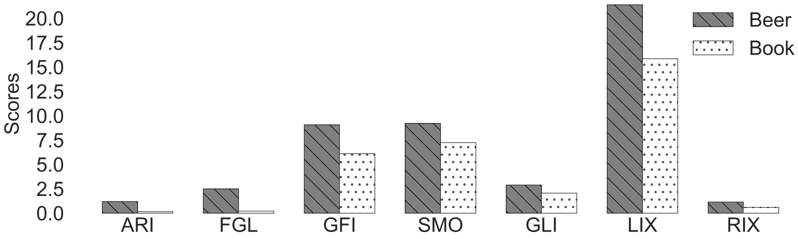


Fig. C.4: Readability Score of Explainations

4 Conclusion

The work provides preliminary results in automatically generating natural language explanations. The model differs from recent works [1, 4], due to the use of attention layer combined with character-level LSTM. The proposed model improves the performance and allow to generate more accurate and trustworthy explanations aligned to the user′s taste.

We would like to improve the model considering: (1) personalizing explanations to benefit the users' preferences based on their expressed sentiments; and (2) testing the model in larger and more varied review domains such as hotels and restaurants.

5 Acknowledgments

This work is supported by Science Foundation Ireland through the Insight Centre for Data Analytics under grant number SFI/12/RC/2289, and Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0).

References

[1] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *Proceedings of the 11th ACM Conference on Recommender Systems*, ser. RecSys'17, 2017, pp. 297–305.

[2] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell, "Explaining the user experience of recommender systems," *User Modeling and User-Adapted Interaction*, vol. 22, no. 4-5, pp. 441–504, 2012.

[3] A. Karpathy, J. Johnson, and F. Li, "Visualizing and understanding recurrent networks," *CoRR*, vol. abs/1506.02078, 2015, international Conference on Learning Representaions.

[4] L. Dong, S. Huang, F. Wei, M. Lapata, M. Zhou, and K. XuT, "Learning to generate product reviews from attributes," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, ser. CECACL'17. Association for Computational Linguistics, 2017, pp. 623–632.

[5] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews," in *Proceedings of*

the 22nd International Conference on World Wide Web, ser. WWW '13. ACM, 2013, pp. 897–908.

[6] S. Maharjan, J. Arevalo, M. Montes, F. A. González, and T. Solorio, "A multi-task approach to predict likability of books," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, ser. EACL'17, vol. 1, 2017, pp. 1217–1227.

Paper D

Neural Explainable Collective Non-negative Matrix Factorization for Recommender Systems

Felipe Costa and Peter Dolog

The paper has been published in the
Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST'18), pp. 35–45, 2018.

Abstract

Explainable recommender systems aim to generate explanations for users according to their predicted scores, the user's history and their similarity to other users. Recently, researchers have proposed explainable recommender models using topic models and sentiment analysis methods providing explanations based on user's reviews. However, such methods have neglected improvements in natural language processing, even if these methods are known to improve user satisfaction. In this paper, we propose a neural explainable collective non-negative matrix factorization (NECoNMF) to predict ratings based on users' feedback, for example, ratings and reviews. To do so, we use collective non-negative matrix factorization to predict user preferences according to different features and a natural language model to explain the prediction. Empirical experiments were conducted in two datasets, showing the model's efficiency for predicting ratings and generating explanations. The results present that NECoNMF improves the accuracy for explainable recommendations in comparison with the state-of-art method in 18.3% *for NDCG@5,* 12.2% *for HitRatio@5,* 17.1% *for NDCG@10, and* 12.2% *for HitRatio@10 in the Yelp dataset. A similar performance has been observed in the Amazon dataset* 7.6% *for NDCG@5,* 1.3% *for HitRatio@5,* 7.9% *for NDCG@10, and* 3.9% *for HitRatio@10.*

1 Introduction

Recommender systems have become an important method in online services, aiming to help users to filter items of their preferences, such as movies, places, or products. The most well-known model to recommend top-N items is collaborative filtering (CF), which recommend items according to the user's similarities with other users and/or items. However, traditionally CF methods focus on users, items, and their interactions to recommend a sorted list of N items.

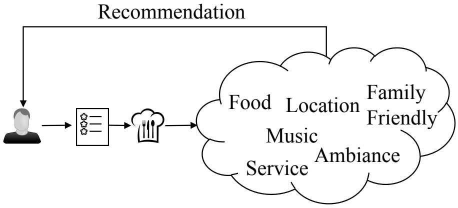


Fig. D.1: Example of Review-aware Recommendation

Among CF techniques, matrix factorization (MF) has been widely applied in recommender systems because of its accuracy in providing personalized recommendation based on user-item interactions, for example, overall ratings. However, users may have different preferences about specific features of the same item as seen in Figure D.1. Hence it is a challenge to infer whether a user would prefer a specific feature from an item based on overall rating. For example, in the restaurant domain, one user may rate a restaurant 3-stars due to service, while another user may give the same rating due to food. This example, shows the importance to model different features when developing a recommendation method, since those features may help to explain *why* a CF model recommends a specific list of items to a user.

Online services provide other explicit feedback besides the overall ratings, such as item's content features, contextual information, sentiments, and reviews. Items' content features describe attributes of an item, for example, which food is served in the restaurant. Contextual information defines the situation experienced by a user, for example, if a user has been in a restaurant for business or leisure. Sentiments describes the positive or negative experience of a user regarding an item. Reviews are user's personal assessment about an item, which may describe items' content features according to her/his preferences. Those explicit feedback have been applied separately as feature to predict user preferences, however as aforementioned, a user may like or dislike an item's content feature in different contexts.

We propose to align those features using non-negative collective matrix factorization model, called NECoNMF. NECoNMF factorizes ratings, item's content features, contextual information, and sentiment in four non-negative low-rank matrices, represented in a common latent space. The hypothesis is that jointly decomposition of different matrices into the same factor space (for example, users' preferences, items' content features, contextual information, and sentiment) improves the prediction of top-N recommendation.

Moreover, the reviews represent an additional information for collaborative filtering. Recent research has modeled reviews to generate explanations for recommender systems, which are known as explainable recommendations. Explicit factor model (EFM) developed by [1] extracts features and user opinions by phrase-level sentiment analysis on user generated reviews, and explain the recommendation based on the extracted information. TriRank proposed by [2] uses a tripartite graph to enrich the user-item binary relation to a user-item-aspect ternary relation. Both projects propose to extract aspects from reviews to generate explainable recommendations, but they do not consider influence maximization in social relation as a source of explanation. Recently, [3] propose the social collaborative viewpoint regression model (sCVR), which detects viewpoints and uses social relations as a latent variable model. sCVR is represented as a tuple of concept-topic-sentiment from both user reviews and trusted social relations.

The aforementioned techniques have shown improvements in explaining recommendations, however they neglected *how* to present the textual explanations while keeping their accuracy in top-N recommendation. Recently, natural language processing techniques have applied deep learning methods, such as recurrent neural network (RNN), which has demonstrated significant improvement in character-level language generation, due to the ability to learn latent information. However, RNN does not capture dependencies among characters, since it suffers from gradient vanishing problem [4]. To solve this issue, long short-term memory (LSTM) have been introduced by [5, 6] revealing good results in generating text upon different datasets [7] and machine translation [8].

In this paper, we propose a neural explainable collective non-negative matrix factorization based on two steps: (1) prediction of user's preferences by collectively factorizing ratings, items' content features, contextual information and sentiments; and (2) generative text reviews given a vector of ratings, which shows specific opinions about different items' features.

The experiments were performed using two real-world datasets: Yelp and Amazon, where our method outperforms the state-of-art in terms of *Hit Ration* and NDCG.

This paper presents the following contributions:

- Neural Explainable Collective Non-negative Matrix Factorization;
- Collective factorization of four matrices: ratings, item's content features, contextual information, and sentiments;

- Text generation model using neural networks;
- Results according to empirical experiments in two real-world datasets.

2 Related Works

Related work can be divided in two tasks: top-N recommendations and explainable recommendations.

Top-N Recommendations. The most well-known methods are Popular Items (PopItem) [9] and PageRank [10], since they present reasonable results and simple implementation. Another widely applied method is matrix factorization (MF), for example SVD [11] and NMF [12]. SVD was first applied on the Netflix dataset, presenting good results in terms of prediction accuracy. Likewise, NMF has received attention, due to its easy interpretability for matrices decomposition.

Matrix factorization has been extended over the years to improve its effectiveness, where collective matrix factorization has shown good performance. This technique was applied in multi-view clustering, where it splits objects into clusters based on multiple representations of the object, as shown by [13, 14]. [13] proposed MultiNMF, using a connection between NMF and PLSA, and [14] proposed a co-regularized NMF (CoNMF), where comment-based clustering is formalized as a multi-view problem using pair-wise and cluster-wise CoNMF.

Local collective embeddings (LCE) is a collective matrix factorization method proposed by [15], which exploits user-document and document-terms matrices, identifying a common latent space to both item features and rating matrix. LCE has shown effectiveness in cold-start problem for news recommendation, however, it has some limitations. The method does not perform well in our domain which covers online services as restaurants and e-commerce, because it uses only two matrices as input and multiplicative update rules as learning model. An extension of LCE is proposed by [16], named CHNMF, where collective factorization is applied in three different matrices using hybrid regularization term. In this paper, we extend the CHNMF approach decomposing a matrix as a product of four matrices: ratings, item's content features, contextual information, and sentiments. Content features are data from each item's metadata, ratings represents user's preferences, contextual information is the situation where the user rates an item, and sentiment is the positive (1) or negative (-1) preference given by a user to an item's feature.

Explainable Recommendations. Explainable recommendation aims to improve transparency, effectiveness, scrutability, and user trust [1]. These criteria can be achieved by different methods to explain the recommendation to a user, such as graph or table, however for this project we selected two baselines which uses textual information to justify their predictions. [1] proposes the explicit factor model based on sentiment lexicon construction and [2] proposed TriRank algorithm to improve the ranking of items for review-aware recommendation.

Our proposal differs from these models in two aspects: we collectively factorize ratings, item's content features, contextual information, and sentiments; and apply a character-level explanation model for recommendation using LSTM for text generation [17].

3 Problem Formulation

The research problem investigated in this paper is defined as followed: *Explain the recommended list of items to each user based on ratings, item's content features, contextual information, and sentiments from user-item interactions*. Modeling the rating data X_u from a set of users U to a set of items I under X_a types of content, X_c types of context, and X_s explicit sentiments as four two-dimensional matrices. We can formally define them as: user-item matrix $X_u = \{x_u \in \mathbb{R}^{u \times i} | 1 \leq x_u \leq 5\}$; user-content feature matrix $X_a = \{x_a \in \mathbb{Z}^{u \times a} | 0 \leq x_a \leq 1\}$; user-context matrix $X_c = \{x_c \in \mathbb{Z}^{u \times c} | 0 \leq x_c \leq 1\}$; and user-sentiment matrix $X_s = \{x_s \in \mathbb{Z}^{u \times s} || x_s| = 1\}$. Where, u is the number of users, i is the number of items, a is the content size, c is the context, and s is the sentiment regarding item features.

Factor models aim to decompose the original user-item interaction matrix into two low-rank approximation matrices. Collective non-negative matrix factorization is a generalization of the classic matrix factorization, where the latent features are stored in four low-rank matrices: ratings as $W \times H_u$; content features as $W \times H_a$; context as $W \times H_c$; and sentiment as $W \times H_s$. Where, H_u denotes a row vector representing the latent features for user u. Similarly, H_a represents the content's latent features a, H_c represents the context's latent features c, and H_s defines the sentiment's latent features. Finally, W represents the common latent space.

The matrix factorization method does not provide human-oriented explanations. To address this particular issue, our model is defined to target the problem of generating natural language review-oriented explanations.

We formulate the explanation for rating prediction as: given input ratings vector $r_{ui} = (r_1, \ldots, r_{|r_{ui}|})$ we aim to generate item explanation $e_i = (w_1, \ldots, w_{|t_i|})$ by maximizing the conditional probability $p(e|r)$. Note, rating r_{ui} is a vector of the user's overall and specific rating of different features from a target item i. While the review t_i is considered a sequence of characters of variable length. For Yelp dataset, we set $|r|$ as 5, representing 5 different features: $food$, $service$, $ambiance$, $location$, and $familyfriendly$. The model learns to compute the likelihood of generated reviews given a set of input ratings. This conditional probability $p(e|r)$ is represented in Equation D.1.

$$p(e|r) = \prod_{s=1}^{|e|} p(w_s|w < s, r) \tag{D.1}$$

where $w < s = (w_1, \ldots, w_{t-1})$.

4 Methodology

In this section, we describe NECoNMF for rating prediction and natural language generation model.

4.1 Collective Matrix Factorization

We propose collective non-negative matrix factorization applied to domains with specific available information, such as, ratings, content features, context, and sentiment. This information is decomposed into four matrices in a common latent space, i.e., each user-item can be related to item's content features, contextual situation, sentiment, and ratings.

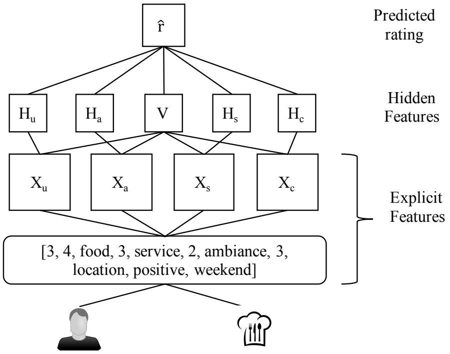


Fig. D.2: Collective Non-Negative Matrix Factorization

Considering an online restaurant service as seen in Figure D.2, where a user may rate a specific restaurant (here defined as an item) according to different criteria, such as, *food*, *service*, *ambiance*, and *location*. Furthermore, the user may write a review and give an overall rating about the item. Based on this example, we may retrieve content-feature matrix X_a, user-item matrix X_u, user-context matrix X_c, and user-sentiment matrix X_s. If we factorize X_a in two low-dimensional matrices, we will observe the content features is belonging to an item. Factorizing X_u leads to find the items according to the users' preferences. While factorizing matrix X_c allow us to identify the hidden contextual factors related to the item. Likewise, factorizing X_s highlight the positive and negative sentiments from a user given to an item's feature. If each matrix is factorized independently it will represent a different latent space and there will be no correlation between content feature, context, sentiment, and rating matrix. The idea of collective non-negative matrix factorization is that each entity: users, items, content features, contexts, and sentiment should be represented in a common latent space, meaning that each item can be described by a set of sentiments (for example, *positive*), and by a set of explicit feedback (for example, *ratings*). The proposal is to collectively factorize X_u, X_a, X_c, and X_s into a low-dimensional representation in a common latent space. The formal definition, is given as the following optimization problem:

$$\begin{aligned} min : f(W) = \frac{1}{2}[\alpha\|X_u - WH_u\|_2^2 + \beta\|X_a - WH_a\|_2^2 \\ +\gamma\|X_c - WH_c\|_2^2 + \omega\|X_s - WH_s\|_2^2 \\ +\lambda(\|W\|_2 + \|H_u\|_2 + \|H_a\|_2 + \|H_c\|_2) + \|H_s\|_2)] \\ s.t. W \geq 0, H_u \geq 0, H_a \geq 0, H_c \geq 0, H_s \geq 0, \end{aligned} \tag{D.2}$$

where the first four terms correspond to the factorization of the matrices X_u, X_a, X_c, and X_s. The matrix W represents the common latent space, and H_u, H_a, H_c, and H_s are matrices representing the hidden factors for each item-user interaction feature. The hyper-parameters are defined by α, β, γ, and ω to control the importance of each factorization with values between 0 and 1. Setting the hyper-parameters as 0.25 gives equal importance to the matrices decomposition, while values of α, β, γ, and ω set as > 0.25 (or < 0.25) give more importance to the factorization of X_u (or X_a, or X_c, or X_s), respectively. The Tikhonov regularization of W is controlled by the hyper-parameter $\lambda \geq 0$ to enforce the smoothness of the solution and avoid overfitting.

Optimization

Alternating optimization is required in NECoNMF to minimize the objective function, since performing collective factorization leads us to find a common low-dimensional space that is optimal for the linear approximation of the user and item data points. Assume the data from user and item has a common distribution where it can exploit a better low-dimensional space, i.e., two data points, u_i and v_j, are close to each other in the low-dimensional space, if they are geometrically close in the distribution. This assumption is known as the *manifold assumption* and is applied in algorithms for dimensionality reduction and semi-supervised learning [15].

We model the local geometric structure through a nearest neighbour graph on a scatter of data points as seen in [15]. Considering a nearest neighbour graph, we represent each data point as node n. Then, we find the k nearest neighbours for each node, and we connect these nodes in the graph. The edges may have: (1) binary representation, where 1 defines one of the nearest neighbours and otherwise 0; (2) or weights representation, for example, pearson correlation. The result of this process is an adjacency matrix A, which may be used to find the local closeness of two data points u_i and v_j.

Based on this assumption, we assume the collective factorization reduces the data point u_i from a matrix X, into a common latent space W as w_i. Then, applying Euclidean distance $\|w_i - w_j\|^2$, we can calculate the distance between two low-dimensional data points and map them into the matrix A. We repeat the process until the stationary point or the established number of maximum iterations, as follows:

$$\begin{aligned} M &= \frac{1}{2}\sum_{i,j=1}^{n} \|w_i - w_j\|^2 A_{ij} \\ &= \sum_{i=1}^{n}(w_i^T - w_i)D_{ii} - \sum_{i,j=1}^{n}(w_i^T - w_i)D_{ii} \\ &= Tr(W^T DW) - Tr(W^T AW) = Tr(W^T LW), \end{aligned} \tag{D.3}$$

where $Tr(\bullet)$ is the trace function, and D is the diagonal matrix whose entries are row sums of A (or column, as A is symmetric). To enforce the non-negative constraints, we need to define the Laplacian matrix as $D_i i = \sum_i A_i j$; $L = D - A$.

We can re-write the optimization problem of function $f(W)$ as:

$$\begin{aligned} min : f(W) = \frac{1}{2}[\alpha\|X_u - WH_u\|_2^2 + \beta\|X_a - WH_a\|_2^2 \\ +\gamma\|X_c - WH_c\|_2^2 + \omega\|X_s - WH_s\|_2^2 \\ +\varphi Tr(W^T LW) \\ +\lambda(\|W\|_2 + \|H_u\|_2 + \|H_a\|_2 + \|H_c\|_2 + \|H_s\|_2)] \\ s.t. W \geq 0, H_u \geq 0, H_a \geq 0, H_c \geq 0, H_s \geq 0 \end{aligned} \tag{D.4}$$

where φ is a hyper-parameter controlling the objective function, and the hyper-parameters α, β, γ, ω, and λ have the same semantics as in Equation D.2.

4.2 Multiplicative Update Rule

Multiplicative update rule (MUR) is applied in the model as regularization term of W. MUR updates the scores in each iteration to reach the stationary point, where we fix the value of W while minimizing $f(W)$ over H_u, H_a, H_c, and H_s. We formalize the partial derivative function in Equation D.5 before calculating the update rules.

$$\begin{aligned}\nabla f(W) = \alpha W H_u H_u^T - \alpha Y_u H_u^T + \beta W H_a H_C^T - \beta Y_a H_a^T \\ +\gamma W H_c H_c^T - \gamma Y_c H_c^T + \omega W H_s H_s^T - \omega Y_s H_s^T + \lambda I_k\end{aligned} \tag{D.5}$$

where k is the number of factors and I_k is the identity matrix with $k \times k$ dimensions.

The update rules are formalized as in Equation D.6, after calculating the derivatives of $f(W)$, $f(H_u)$, $f(H_a)$, $f(H_c)$, and $f(H_s)$ from Equation D.5.

$$W = \frac{[\alpha Y_u H_u^T + \beta Y_a H_a^T + \gamma Y_c H_c^T + \omega Y_s H_s^T]}{[\alpha H_U H_U^T + \beta H_a H_a^T + \gamma H_c H_c^T + \omega H_s H_s^T + \lambda I_k]} \tag{D.6}$$

where $\frac{\bullet}{\bullet}$ corresponds to left division.

The results of each iteration give us the solution for pair-wise division, where the objective function and the delta decrease on each iteration of the above update rule, guaranteeing the convergence into a stationary point. Note, we map any negative values from W matrix to zero, becoming non-negative after each update.

4.3 Barzilai-Borwein

The Brasilai-Borwein regularization term is used to optimize the hidden factor matrices H_u, H_a, H_c, and H_s. We represent the hidden factor matrices as H for the input matrices X_u, X_a, X_c, and X_s, since they present equal problem as shown in Equation D.7:

$$\min_{W \geq 0} : f(H) = \frac{1}{2} \|X - WH\|_F^2 \tag{D.7}$$

We map all the negative values into zero through $P(\bullet)$ for any $\alpha > 0$, as we defined for W in the previous section. Equation D.8 formally describes the statement as:

$$\|P[H - \alpha \nabla f(H)] - H\|_F = 0. \tag{D.8}$$

Applying ϵ_H in Equation D.8 lead us to the following regularization term $\|P[H - \alpha \nabla f(H)] - H\|_F \leq \epsilon_H$, where $\epsilon_H = max(10^{-3}, \epsilon)\|P[H - \alpha \nabla f(H)] - H\|_F$. We decrease the stopping tolerance by $\epsilon = 0.1\epsilon_H$, if the Borzilai-Borwein algorithm defined in [16] solves Equation D.7 without any iterations.

The gradient $\nabla f(W)$ of $f(H)$ is Lipschitz term with constant $L = \|W^T W\|_2$. L is not expensive to obtain, since Equation D.4 defines $W^T W$ with dimensions $k \times k$ and $k \ll min\{m, n\}$. For a given $H_0 \geq 0$:

$$\mathcal{L}(H_0) = \{H | f(H) \leq f(H_0), H \geq 0\}. \tag{D.9}$$

Following the definition of Equation D.9 we have the stationary point of the Barzilai-Borwein method.

4.4 Top-N Recommendation Process

Factorizing the input matrices return the trained matrices W, H_u, H_a, H_c, and H_s allows us to use the hidden factor elements for prediction. Given the new items' vector v_i, we can predict the most preferable items according to the user's interest v_u. To solve this issue we project the items vector v_i to the common latent space by solving the overdetermined system $v_i = wH_u$ using the least squares method. The vector w, captures the factors, in the common latent space, that explain the preferable items v_i. Then, by using the low-dimensional vector w we infer the missing part of the query: $v_u \leftarrow wH_t$ where H_t is the concatenation of content feature, context, and sentiment matrices $H_t = H_a||H_c||H_s$. Each element of v_u represents a score of how likely the user will like a new item. Then, given these scores, we may sort the list of items.

Moreover, given the sorted list of items and their rating scores, we explain the predicted ratings using the natural language generation model. The explanation model uses the ratings as attention model to generate personalized sentences according to user's writing style. The training model identifies positive and negative sentences according to the user's previous reviews.

4.5 Natural Language Explanation

The natural language generation model is divided in (1) four sub-models: context encoder, LSTM network, attention layer, and generator model; and (2) two input sources: review text and concatenated character embeddings of user, item, and rating vector, as presented in Figure D.3. First, users and items embeddings are learned from *doc2vec* model [18], where characters of reviews are converted to one-hot vectors, corresponding to the input time-steps of LSTM network. Second, the embeddings are concatenated with the outputs of LSTM, which are inputs for the following attention layer. Finally, the generator model produce sentences using outputs from the attention layer.

Context encoder. It aims to encode the input character into one-hot encoding, then concatenate the ratings given by a user according to different item's content features. This data will later become the input for our LSTM network as seen in Fig. D.3. To do so, we created a dictionary for the characters in the corpus to define their positions. The dictionary is used to encode the characters during the training step and to decode during the generating step. One-hot encoding vector is generated for each character in the reviews based

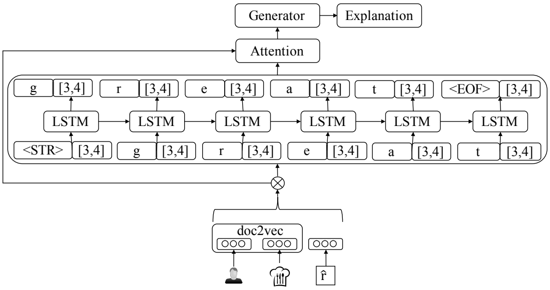


Fig. D.3: Natural Language Explainability model

on its position in the dictionary. Furthermore, the vector is concatenated with a set of ratings varying from 0 to 1, as shown in Equation D.10.

$$X'_t = [onehot(x_{char}); x_{aux}] \tag{D.10}$$

LSTM network. LSTM is an extended version of recurrent neural network (RNN), where neurons transmit information among other neurons and layers simultaneously, as presented in Figure D.3. Therefore, LSTM is built upon a sequential connection of forget, input and output gates. The forget gate aims to decide which old information should be forgotten; the input gate updates the current cell state; and the output gate selects the information to go to the next layer and cell. First, the LSTM network receives the input data x_t at time t and the cell state C_{t-1} from previous time step $t-1$. Second, the input data feed the forget gate, where it chooses which information will be discarded. In Equation D.11, f_t defines the forget gate in time t, where W_f and b_f refers to the weight matrix and bias, respectively. The third step is to define which information should be stored in cell state by the input gate i_t. During the fourth step, the cell creates a candidate state C'_t by a *tanh* layer. We update the current state C_t according to the candidate state, the previous cell state, the forget gate, and the input gate. During the final step, the data goes to the output gate, where it uses *sigmoid* function layer to define the output and multiply the *tanh* with the current cell state C_t to return the next character with the highest probability.

$$\begin{aligned}
f_t &= \sigma([x_t, C_{t-1}] \odot W_f + b_f) \\
i_t &= \sigma([x_t, C_{t-1}] \odot W_i + b_i) \\
C'_t &= \tanh([x_t, C_{t-1}] \odot W_c + b_c) \\
C_t &= f_t \odot C_{t-1} + i_t \odot C'_t \\
o_t &= \sigma([x_t, C_{t-1}] \odot W_o + b_o) \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned} \tag{D.11}$$

Attention layer. The attention layer adaptively learns soft alignments h_t between character dependencies c_t and *attention* inputs. Equation D.12 formally describes the character dependencies using attention layer $h_t^{attention}$ as explained by [19].

$$\begin{aligned} c_t &= \sum_{i}^{attention} \frac{\exp(\tanh(W_s \odot [h_t, attention_i]))}{\sum \exp(\tanh(W_s \odot [h_t, attention_i]))} attention_i \\ h_t^{attention} &= \tanh(W_1 \odot c_t + W_2 \odot h_t) \end{aligned} \tag{D.12}$$

Generator model. The generator model is built on character-level. To do so, we have to maximize a non-linear *softmax* function to compute the conditional probability p among the characters, as presented in Equation D.13. The text generation task concatenates an initial *prime* text as the *start* symbol in each generated review-based explanation according to different item's content features. Finally, the network feeds the *softmax* layer with its output data, as shown in Equation D.13.

$$p = \text{softmax}(H_t^{attention} \odot W + b), \quad char = \arg\max p \tag{D.13}$$

where $H_t^{attention}$ is the output of a LSTM cell, W and b are the weight and bias of the *softmax* layer, respectively.

This procedure produces a character *char* recursively for each time step, until it finds the pre-defined *end* symbol.

5 Experiments

In this section, we describe our experiment setup. We start by detailing the datasets, the evaluation metrics and the baselines.

5.1 Datasets

We use two datasets to conduct our experiments: Yelp and Amazon.

The original Yelp dataset was published in April 2013 and has been used by the recommender systems community, due to the user reviews. The dataset has 45,981 users, 11,537 items, and 22,907 reviews, however it is sparse, since 49% of the users have only one review, making it difficult to evaluate in top-N recommendation. We filtered the users with more than 10 reviews as [1, 2]. The Amazon dataset contains more than 800k users, 80k items and 11.3M reviews, however it is more sparse than Yelp, making us adopt the same strategy as applied in the previous dataset. Table D.1 summarizes the dataset information.

	Yelp	Amazon
#users	3,835	2,933
#items	4,043	14,370
#reviews	114,316	55,677
#density	0.74%	0.13%

Table D.1: Dataset Summarization

The experiments were performed in a Unix server with the following settings: 32GB of RAM and 8 core CPU Intel Xeon with 2.80GHz. The parameters for collective matrix factorization were set as: *learning rate* = 0.001; k = 45; *iterations* = 50 (to ensure the convergence point); and λ = 0.5. The LSTM network has two hidden layers with 1024 LSTM cells per layer, where the input data was split in 100 batches with size equal to 128 and each batch has a sequence length equal to 280. For this experiment, we applied cross-validation to avoid overfitting, where each dataset were divided into $5 - folds$.

5.2 Evaluation Metrics

The output of the algorithms is a ranked list of items considering the user's feedback. For this list, the effectiveness of the top-N recommendation is measured according to information retrieval ranking metrics: NDCG and *Hit Ratio* (HR). We apply NDCG to measure the ranking quality as defined by Equation D.14, where the metric gives higher score to the items at the top rank, and lower scores to the items at the low rank.

$$NDCG@N = Z_N \sum_{i=1}^{N} \frac{2^{r_i} - 1}{log_2(i+1)} \tag{D.14}$$

where Z_N normalizes the values, which guarantees the perfect ranking has a value of 1; and r_i is the graded relevance of item at position i. We define $r_i = 1$ if the item is in the test dataset, and otherwise 0.

Hit Ratio has been commonly applied in top-N evaluation for recommender systems to asses the ranked list with the ground-truth test dataset (GT). A *hit* is denoted when an item from the test dataset appears in the recommended list. We calculate it as:

$$Hit@N = \frac{NumberofHits@N}{|GT|} \tag{D.15}$$

The number of listed item in the trained dataset is defined by N. Therefore, we set $N = 5$ and $N = 10$, due to large values of N would result in extra work for the user to filter among a long list of relevant items. High values of NDCG or HR indicate better performance.

5.3 Comparison Baselines

We list the related works in Section 2, where we highlighted the techniques based on their recommendation tasks: top-N and explainable recommendations. In this section we describe four baselines, where two are popular techniques for top-N recommendations and the two other are recent techniques applying review-based explanations.

Top-N Recommendation

- *ItemPop.* The method ranks items by their popularity given by the number of ratings. It usually presents good performance, because users tend to consume popular items.
- *PageRank.* Personalized Pagerank is a widely used method for top-N recommendation. We applied the configuration proposed by [2], considering the user-item graph and the damping parameter is respectively optimized to 0.9 and 0.3 for Yelp and Amazon datasets.

Explainable Recommendation

- *EFM.* The model applies a phrase-level sentiment analysis of user reviews for personalized recommendation. It extracts explicit product features and user opinions from reviews, then incorporates user-feature and item-feature relations as well as user-item ratings into a hybrid matrix factorization framework. The method has k as hyper-parameter to define the number of most cared features, which we define $k = 45$ as reported in [1] to have the best performance in the Yelp dataset.
- *TriRank.* The model ranks the vertices of user—item—aspect tripartite graph by regularizing the smoothness and fitting constraints. TriRank is used for review-aware recommendation, where the ranking constraints directly model the collaborative and aspect filtering. The authors describe in [2] an experiment regarding the parameters used by TriRank algorithm, where setting $item - aspect$, $user - aspect$, and $aspect - query$ as 0 results in poor performance. For the experiments in this paper we use the default settings $\alpha = 9$, $\beta = 6$, and $\gamma = 1$.

6 Results and Discussions

In this section we present the results regarding the rating prediction for top-N recommendation, and discuss the explanations for rating prediction.

Dataset	**Yelp**				**Amazon**			
Metric	**NDCG@5**	**HIT@5**	**NDCG@10**	**HIT@10**	**NDCG@5**	**HIT@5**	**NDCG@10**	**HIT@10**
Top-N Algorithms								
ItemPop	0.0110	0.0136	0.0185	0.0306	0.0077	0.0082	0.0136	0.0238
PageRank	0.0235	0.0278	0.0313	0.0452	0.0978	0.1070	0.1029	0.1200
Explainable Recommendation								
TriRank	0.0258	0.0313	0.0353	0.0527	0.1033	0.1127	0.1086	0.1266
EFM	0.2840	0.0448	0.2955	0.0678	0.3615	0.1284	0.3670	0.1429
NECoNMF	**0.3366**	**0.0503**	**0.3461**	**0.0763**	**0.3892**	**0.1301**	**0.3962**	**0.1486**

Table D.2: NDCG and Hit Ratio results for NECoNMF and compared methods at rank 5 and 10

6.1 Overall Performance

This section describes the overall performance of NECoNMF regarding the rating prediction for top-*N* recommendation in comparison with other methods.

Analyzing the NDCG@5 score of NECoNMF in comparison with ItemPop in Table D.2 we observe an improvement of 30× for the Yelp dataset and 50× for the Amazon dataset, while HIT@5 is 4× better for the Yelp dataset and 16× for the Amazon dataset. ItemPop has shown the poorest overall performance because it does not apply personalization during its prediction task.

On the other hand, personalized PageRank had a good performance due to apply the user's historical information to predict the user's preferences. Despite the good performance, PageRank does not use latent information to infer the recommendation, for example item's content features, context, sentiment, and ratings, what may explain the lower performance when compared to NECoNMF. Comparing the results for the Yelp dataset the accuracy of NECoNMF compared to PageRank was improved in 14.4× for NDCG@5 and 2× for HIT@5. Applying the recommendation model for the Amazon dataset leads us to a similar result where NECoNMF performs better in 3.9× for NDCG@5 and 21% for HIT@5.

TriRank performed poorer in comparison to NECoNMF, because it applies tripartite graph approach to predict users' preferences making it unable to identify hidden features. During the experiments for the Yelp dataset the accuracy was improved in 13× for NDCG@5 and 60% for HIT@5. Likewise, for the Amazon dataset we observed an improvement of 3.7× for NDCG@5 and 15.4× for HIT@5. Observing *Hit Ratio* scores in Table D.2 we noticed PageRank and TriRank have close scores, however TriRank performed better because it applies tripartite graph user–item–aspect, allowing TriRank to incorporate one more feature during the recommendation process.

Comparing NECoNMF and EFM, we observe a close performance between them, however NECoNMF presented an improvement for top-*N* recommendation. Considering the results in Table D.2 we observe an improvement of 18.3% in NDCG@5, 12.2% in HIT@5, 17.1% in NDCG@10, and 12.2%

in HIT@10 for the Yelp dataset. Similar improvement was observed in the Amazon dataset, 7.6% in NDCG@5, 1.3% in HIT@5, 7.9% in NDCG@10, and 3.9% in HIT@10. NECoNMF factorizes item's content features and contextual information as additional features allowing a better user modeling in the vectorial space and improving the personalized recommendation task. On the other hand, EFM models the user's behaviour based only on ratings and aspects. Those features play an important role in achieving better NDCG score, hence it defines users' interests in specifics scenarios, for example, brunch in a restaurant on a Sunday. Furthermore, EFM factorizes its input matrices into four different latent spaces, while NECoNMF collectively factorizes into one common latent space among the matrices. Therefore, jointly factorizing the matrices may identify hidden latent features not selected by EFM model during the recommendation process.

Moreover, the models presented better performance for top@10 than top@5 recommendation, due to the error is minimized when the system has a higher number of items. Analyzing Figures D.4 and D.5 we observe NECoNMF presents higher accuracy performance for top-N recommendation when N varies from 5 to 50.

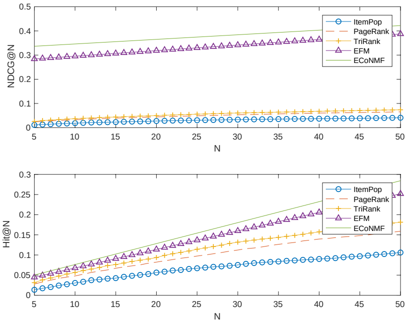


Fig. D.4: Empirical evaluation on Yelp dataset for N from 5 to 50

6.2 Explainability

NECoNMF presents the ability to provide explainable recommendation, however explainability for recommendation is not easy to evaluate. We apply the evaluation method described by [1], where they assess the quality of explanations throughout examples generated by the explainable model.

Table D.3 shows three explanations outcomes according to different predicted user's ratings. NECoNMF presents an explanation according to user's writing style for each tuple user-item-rating, where the dark grey represents negative sentiment, and the light grey describes positive sentiment regard-

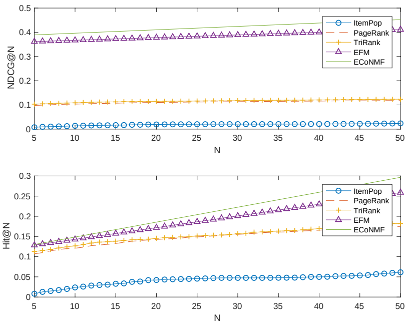


Fig. D.5: Empirical evaluation on Amazon dataset for N from 5 to 50

User	Item	Rating	Generated Explanation
A	X	1	i was disappointed . it was too small and noisy . food was good , but i just can't come back with my family .
A	Y	3	this was a nice restaurant . i liked the service and the location . i'm not sure i'd recommend the food there.
B	X	5	i loved this restaurant . i could not leave it. i loved the service and the food . i will be back again next weekend .

Table D.3: Example explanations produced by NECoNMF on the Yelp dataset.

ing different item's features. Furthermore, the mid-grey scale represents the contextual feature such as *family*, *location* and *weekend*. Note, the generated explanation is presented in a personalized item's review style, because reviews have a high influence in user's decision. Comparing to [1] explanations, we observe a better readability in Table D.3 when using review-based explanation, due to natural language text generation.

7 Conclusions

We proposed a neural explainable collective non-negative matrix factorization (NECoNMF) for recommender systems combining ratings, content features, sentiment, and contextual information in a common latent space. Furthermore, we introduced a neural explainable model to interpret the predicted top-N recommendation. Finally, we presented the results regarding the experiments in two datasets, where we observed that NECoNMF outperforms the state-of-the-art methods.

The top-N recommendation task was addressed using four different matrices as input for collective non-negative matrix factorization. The combination of content features, contexts, ratings, and sentiment play an important role in explaining the recommended list of items to a user.

The explainable model proved to be effective for the review-oriented explanation task. The generated explanations may help users during their decision regarding specific item's features, as users tend to trust in the review-based explanation. Moreover, the character-level text generation has the benefit of generating readable personalized text.

We would like to improve the readability presented by the explainable model and further extend the project into a general explainable recommender system, which is able to explain any recommendation method. Furthermore, investigate technical improvements related to the cold-start problem.

8 Acknowledgments

The authors wish to acknowledge the financial support and the fellow scholarship given to this research from the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0).

References

[1] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, ser. SIGIR '14. New York, NY, USA: ACM, 2014, pp. 83–92.

[2] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1661–1670.

[3] Z. Ren, S. Liang, P. Li, S. Wang, and M. de Rijke, "Social collaborative viewpoint regression with explainable recommendations," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17. New York, NY, USA: ACM, 2017, pp. 485–494.

[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] S. El Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," in *Proceedings of the 8th International Conference on Neural Information Processing Systems*, ser. NIPS'95. Cambridge, MA, USA: MIT Press, 1995, pp. 493–499.

[7] A. Karpathy, J. Johnson, and F. Li, "Visualizing and understanding recurrent networks," *CoRR*, vol. abs/1506.02078, 2015, international Conference on Learning Representaions.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[9] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 39–46.

[10] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th International Conference on World Wide Web*, ser. WWW '02. New York, NY, USA: ACM, 2002, pp. 517–526.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[12] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems*, ser. NIPS'00. Cambridge, MA, USA: MIT Press, 2000, pp. 535–541.

[13] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proc. of Intl. Conf. on Data Mining*. SIAM, 2013, pp. 252–260.

[14] X. He, M.-Y. Kan, P. Xie, and X. Chen, "Comment-based multi-view clustering of web 2.0 items," in *Proceedings of the 23th International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: ACM, 2014, pp. 771–782. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567975

[15] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: ACM, 2014, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/2645710.2645751

[16] F. Costa and P. Dolog, "Hybrid learning model with barzilai-borwein optimization for context-aware recommendations," in *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida USA., May 21-23 2018.*, 2018, pp. 456–461.

[17] F. Costa, S. Ouyang, P. Dolog, and A. Lawlor, "Automatic generation of natural language explanations," in *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, ser. IUI'18. ACM, 2018, pp. 57:1–57:2.

[18] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1188–II–1196.

[19] L. Dong, S. Huang, F. Wei, M. Lapata, M. Zhou, and K. XuT, "Learning to generate product reviews from attributes," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, ser. CECACL'17. Association for Computational Linguistics, 2017, pp. 623–632.

Paper E

Collective Embedding for Neural Context-Aware Recommender Systems

Felipe Costa and Peter Dolog

The paper is under revision for
a Conference publication

Abstract

Context-aware recommender systems consider contextual factors (for example, time, location, or the company of other people) as additional information to predict user's preferences. Among the contextual information, time is an important feature because user preferences tend to be similar in the near future. Researchers have modeled temporal features through Tensor Factorization (TF). However, TF models suffer from incapability of capturing specific temporal patterns. To address this limitation observed in previous works, we propose Collective embedding for Neural Context-Aware Recommender Systems (CoNCARS). The model jointly represents the item, user and time embeddings. Then, CoNCARS use the pairwise product to model the user-item-time correlations between dimensions of the embedding space. The hidden features feed our Convolutional Neural Network (CNN) to learn the non-linearities between different features. Finally, we combine the output from our CNN in the fusion layer and then predict the user's preference score. We conduct extensive experiments on real-world datasets, which demonstrate that CoNCARS improves the top-N item recommendation task and outperform the state-of-the-art recommendation methods.

The layout has been revised.

1 Introduction

Recommender systems algorithms aim to mitigate the information overload phenomenon, where users have difficulty to decide when facing an overwhelming amount of choices. Recommender systems play an important role in filtering out irrelevant information and selecting only a personalized subset of items to maximize the users' satisfaction. Collaborative Filtering is often the recommendation technique applied to the modern systems, exploiting previous user-items interactions and the interactions of other users for modeling user's preference. Among the various CF methods, model-based CF, for example, matrix factorization based methods are known to outperform other methods and have become the state-of-art solution of recommendation research.

The design of a CF model requires the understanding of how to represent a user and an item, and how to model the user-item interaction based on their representation. Matrix factorization (MF) characterizes users and items as vectors of latent factors (also known as embeddings) and represents a user-item interaction as the inner product of the user and item embeddings as the preference score. Several extensions have been developed for MF to improve both the modeling and learning aspect. For example, APR [1] learns user and item embeddings through an adversarial training procedure, NeuMF [2] extends MF by learning embeddings with a multi-layer perceptron network, and BPR [3] optimizes MF for implicit feedback applying pairwise ranking method.

Despite the accuracy of the previous methods, they assume that the dimensions of the embedding space are independent with each other. This hypothesis is impractical since the embedding dimensions could contain dependent features such as items' attributes. Furthermore, this hypothesis is sub-optimal for learning from real-world feedback data that is rich in contextual information as demonstrated by context-aware models [4, 5], which outperformed other models by learning the interaction function from the data.

Among the context-aware models for CF, time-aware recommender systems have gained some attention, since users preferences remain the same in the near future [6]. Tensor Factorization (TF) provides the state-of-art performance by projecting users and items embeddings into a latent space with the time embeddings [5]. We argue that a potential limitation of this method is that TF assumes two consecutive time slots are independent, which makes it infeasible to predict for the next time slot. Moreover, TF does not capture the time-evolving patterns, for example, the user's impression to a movie may be dynamically affected by its winning of some movie awards or if the user's preferences are similar over time. Figure E.1 illustrates a scenario where a user watches a sequence of movies in a different time slot, and after that, the recommender engine suggests a list of items according to the previous sequence.

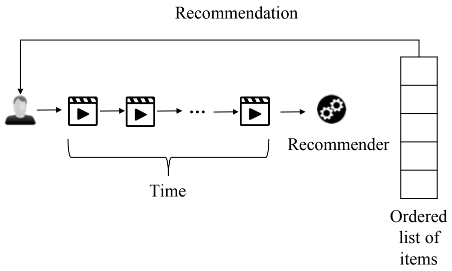


Fig. E.1: Example for user behaviour over time.

To address these limitations, we propose a new architecture for Collective embedding for Neural Context-aware Recommender Systems (CoNCARS). CoNCARS represents a user by the embeddings of all items interacted with the user in a specific time-slot. Likewise, CoNCARS models the item by the embeddings of all users interacted with the item in a specific time-slot.

We incorporate the time as a feature to identify the importance of different interacted items. To do that, we compute the time-based user embeddings by aggregating the item and time embeddings. Similarly, we compute the time-based item embeddings by aggregating the user and time embeddings. The collective embeddings related to each user and item jointly project their latent representations more accurately as shown in our empirical evaluation.

We also combine the time-based item embeddings and time-based user embeddings with the user and item embeddings to model the pairwise user-item correlations. Moreover, CoNCARS learn the non-linear correlations between different dimensions with the Convolutional Neural Network (CNN).

To summarize, this paper presents the following contributions:

- A context-aware model named CoNCARS to improve the prediction based on user's preferences over time;
- A training model for a better learning correlation between users, items and time embedding dimensions;
- Empirical evaluation in three real-world datasets demonstrating the effectiveness of the CoNCARS model.

2 Related Works

Context-aware recommender systems improved the recommendation task, especially by minimizing the cold-start problem and by retrieving items based on contextual information. Time has shown to be strong contextual information because users tend to change or keep their behavior over time [6, 7]. For a better overview, we divide the time-based recommendations approaches into two categories and exploit the techniques closely related to ours:

- *Time-dependent recommender systems* model user's preference as a chronological sequence. It considers the order of the events to have a strong influence on the estimated predictions. This approach requires that the input data is in chronological order and does not take the exact timestamps into account. Therefore, the time-dependent approach aims to adapt the recommendations according to changes over time.

- *Time-aware recommender systems* assume the time as a contextual feature during the training step, aiming to model the time as being cyclic. Timestamps are used as additional information to enrich the model. Time-aware methods consider that a user's behavior repeats at regular time intervals. Based on this assumption, time-aware recommender systems propose to have more accurate predictions of similar patterns in the future.

In this paper, we propose a time-aware recommender system. Usually, the time-aware recommender system is modeled in three different ways: pre-filtering, post-filtering, and contextual modeling. Pre- and post-filtering have shown slightly lower performance when comparing to contextual modeling. Therefore, we explore the contextual modeling technique.

Baltrunas *et al.* [8] propose the Context-aware Matrix Factorization (CAMF) by extending matrix factorization using context as a baseline predictor to represent the interaction of contextual information with users or items. Another variation of MF is called TF, considering time as a third-dimensional factor in the user-item interaction [5, 9]. CMF is an extension of MF proposing to collective factorize multiple representations of user-item relation, identifying a common latent space to both item features and user-item interaction vector [4, 10–12].

Advances in deep learning research field allowed improvements on the recommender systems research area, especially in learning non-linear users-items correlations. He *et al.* [2] provide a neural collaborative filtering model, named NeuMF, which adds an adaptable multi-layer perceptron (MLP) to learn the interaction function. Wu *et al.* [13] propose an autoencoder based CF method to improve the effectiveness for top-N recommendation task. Kim *et al.* [14] integrate Convolutional Neural Network (CNN) into a Probabilistic Matrix Factorization (PMF) model to address the sparsity problem in recommendation techniques. Nonetheless, the deep collaborative filtering approaches do not exploit the advantages of time as an important dimension for the recommendations task.

Our recommendation method differentiates from the methods described above by capturing the temporal information provided by user-item interactions. Our model represents a user-based embedding for each item considering the time as a contextual feature, which identifies the importance of different users and items. Likewise, our model denotes an item-based embedding for each user according to the time to identify the relevance of each user and item.

3 Problem Formulation

Consider a scenario of a user **u** interacting with an item **i** in time **t** in online services, for example, Yelp, Pinterest or Netflix. To model the relationships among users and items over time, we apply collective embedding to represent their interactions.

The set of users is denoted as $U = \{u_1, ..., u_M\}$ and the set of items as $I = \{i_1, ..., i_N\}$. $R = \{r_{ui} \in \mathbb{R}^{M \times N} | 1 \leq r_{ui} \leq 5\}$ denotes the rating matrix, where r_{ui} is the rating of user **u** on item **i**. r_{ui} is labeled as *unk* if user **u** did not interact with item **i**. The matrix R with implicit feedback is modeled as:

$$R_{ui} = \begin{cases} 0, & \text{if } R_{ui} = unk \\ 1, & \text{otherwise} \end{cases} \tag{E.1}$$

Recommender systems solve the problem of predicting the preference score of each unobserved entry in R. The preference scores determine the order of items displayed in the recommended list of items. We generate the scores according to Equation E.2.

$$\hat{r}_{vj} = F(u_v, i_j | \mathbf{\Theta}) \tag{E.2}$$

where $\hat{r}_{vj}$ is the predicted score of interaction r_{vj} between user $\mathbf{u}_v$ and item $\mathbf{i}_j$, $\mathbf{\Theta}$ represents the model parameters, and F is the function which estimates the predicted scores based on $\mathbf{\Theta}$. The function F is used to find an optimal list of items for an individual user according to her/his preferences.

Traditionally, MF methods define function F based on element-wise product of $\mathbf{p}_u$ and $\mathbf{q}_i$ to predict $\hat{R}_{vj}$ as demonstrated by [15], where $\mathbf{p}_u$ and $\mathbf{q}_i$ defines the hidden latent factors of $\mathbf{u}_v$ and $\mathbf{i}_j$, respectively.

$$\hat{R}_{vj} = F(u_v, i_j | \Theta) = \mathbf{p}_v^T \mathbf{q}_j \tag{E.3}$$

However, CoNCARS uses pairwise product to calculate the interactions between users and items over time. Formally, four sets are identified: $\mathbf{u} = (u_1, u_2, \ldots, u_{n-1}, u_n)$, $\mathbf{i} = (i_1, i_2, \ldots, i_{m-1}, i_m)$, $\mathbf{it} = (it_1, it_2, \ldots, it_{h-1}, it_h)$, and $\mathbf{ut} = (ut_1, ut_2, \ldots, ut_{h-1}, ut_h)$. The entry u_1 in the set $\mathbf{u}$ denotes the interactions between user **u** and item i_1, where $u_1 = 1$ if the user **u** interacted with item i_1 and $u_1 = 0$ if not. The entry i_1 in the set **i** represents the interactions between item **i** and user u_1, where $i_1 = 1$ if the item **i** interacted with user u_1 and $i_1 = 0$ otherwise. The entry it_1 in the set **it** records the time **t** when the item **it** was consumed by user $\mathbf{it_1}$. Similarly, the entry ut_1 in the set **ut** records the time **t** when the user **ut** interacted with the item $\mathbf{ut_1}$. Assuming, the user preferences may repeat over time, a discretization on time is performed to avoid the set from being extremely sparse. The time is divided in **T** intervals in total. $it_1 = 1$ if the item **i** is consumed in the time interval d and $it_1 = 0$ if not. The same is applied on the set **ut**.

We aim to solve the following problem. Given the four sets mentioned above, find an embedding model which will be able to capture both, direct interactions between the dimensions and non-linear interactions, and provide an accurate time-aware recommendation of items.

4 Proposed Model

CoNCARS has multiple layers as illustrated in Figure E.2, which defines the prediction model according to Equation E.2. We describe the detailed architecture of the CoNCARS model in the following sections.

4.1 Input Layer

Considering the model as a classifier, we aim to determine whether the user-item pair $(\mathbf{u}, \mathbf{i})$ should have a higher score than $(\mathbf{u}, \mathbf{j})$ for specific time **t**. We define one-hot representation vectors **P** for user **u** and **Q** for item **i**. In addition to the one-hot vectors, CoNCARS combines the binary interaction vectors **X** and **Y** from the observed interactions for **u** and **i** in time **t**. This, gives us four feature vectors for both **u** and **i** from the input layer.

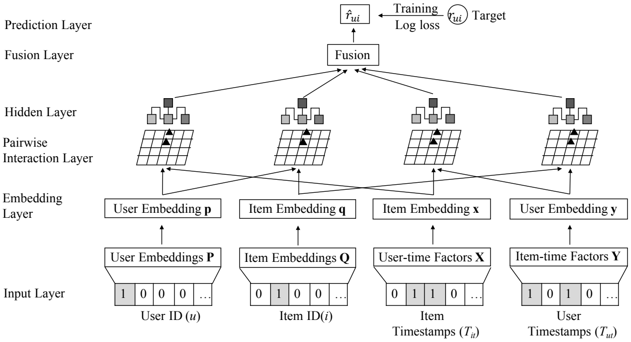


Fig. E.2: Collective Embedding for Neural Context-aware Recommender Systems.

4.2 Collective Embedding Layer

This layer aims to embed meaningful information regarding the user-item interaction. CoNCARS considers the user-item, item-user, user-time, and item-time vectors interactions. Formally, the Equation E.4 defines each user feature vector and Equation E.5 defines each item feature vector from the input layer as an embedding vector.

$$\mathbf{I}_{p_u} = \mathbf{P}^T \mathbf{u} \tag{E.4}$$

$$\mathbf{I}_{q_i} = \mathbf{Q}^T \mathbf{i} \tag{E.5}$$

where $\mathbf{P} \in \mathbb{R}^{U \times K}$ denotes the user embedding matrix and $\mathbf{Q} \in \mathbb{R}^{I \times K}$ denotes the item embedding matrix. K defines the number of hidden factors.

The interaction vectors $\mathbf{X}$ and $\mathbf{Y}$ are considered as another group of latent factors associated with temporal dynamics $\mathbf{x}_u$ for user u and $\mathbf{y}_i$ for item i, respectively. Consider, the user $\mathbf{u}$ interacted with item $\mathbf{i}$ in different times $\mathbf{t}$, formally defined as:

$$\mathbf{I}_{y_u} = \frac{\sum_{i \in T(u)} \mathbf{f}_i}{\sqrt{|T(u)|}} \tag{E.6}$$

$$\mathbf{I}_{x_i} = \frac{\sum_{u \in T(i)} \mathbf{g}_u}{\sqrt{|T(i)|}} \tag{E.7}$$

where $\mathbf{I}_{y_u}$ is the time-based user embedding considering the $\mathbf{f}_i$ latent factors and $\mathbf{I}_{x_i}$ is the time-based item embedding considering the $\mathbf{g}_u$ latent factors. $\mathbf{T}(\mathbf{u})$ and $\mathbf{T}(\mathbf{i})$ denotes all positive entries in $\mathbf{X}$ and $\mathbf{Y}$, respectively. Assuming that distinct items may interfere in distinct user's preference, we rewrite user-item interaction as:

$$\mathbf{I}_{y_u} = \sum_{i \in T(u)} \alpha_i \mathbf{f}_i \tag{E.8}$$

$$\mathbf{I}_{x_i} = \sum_{u \in T(i)} \alpha_u \mathbf{g}_u \tag{E.9}$$

where α_i denotes the importance degree for item $\mathbf{i}$ rated by the user $\mathbf{u}$ at time $\mathbf{t}$ and α_u denotes the importance degree for user $\mathbf{u}$ interaction with item $\mathbf{i}$ at time $\mathbf{t}$. We define the importance degree for item $\mathbf{i}$:

$$\mathbf{h}_i = tanh(\mathbf{W_c} \mathbf{f}_i + \mathbf{b}_a) \tag{E.10}$$

$$\alpha_i = \frac{exp(\mathbf{h}_i^T \mathbf{h}_a)}{\sum_{i \in T(u)} exp(\mathbf{h}_i^T \mathbf{h}_a)} \tag{E.11}$$

where $\mathbf{W_c}$ denotes the weight matrix, $\mathbf{b}_a$ denotes the bias vector, and $\mathbf{h}_a$ is the time contextual vector. First, we give the item factor $\mathbf{f}_i$ as the input of one convolutional layer that produces $\mathbf{h}_i$ as the latent representation of $\mathbf{f}_i$. Then, we compute the similarity between $\mathbf{h}_i$ and $\mathbf{h}_a$, resulting in a normalized importance degree α_i using a softmax function. We employ a similar computation given by Equation E.10 and E.11 to calculate the time-based user representation $\mathbf{I}_{x_i}$.

4.3 Pairwise Interaction Layer

The output from the embedding layer feeds the pairwise interaction layer to model correlations between user $\mathbf{u}$ and item $\mathbf{i}$ in time $\mathbf{t}$. We model the user-item interactions considering the temporal context and obtain four representation vectors as formally defined in Equation E.12.

$$\mathbf{e}^n = \phi_L^n(...(\phi_2^n(z_0[n]))...) \tag{E.12}$$

$$\phi_l^n = \sigma_l^n(\mathbf{W}_l^n \mathbf{z}_{l-1}^n + \mathbf{b}_l^n), \quad l \in [1, L] \tag{E.13}$$

$$\mathbf{z}_0 = [\mathbf{p}_u \otimes \mathbf{x}_i, \mathbf{p}_u \otimes \mathbf{q}_i, \mathbf{y}_u \otimes \mathbf{x}_i, \mathbf{y}_u \otimes \mathbf{q}_i] \tag{E.14}$$

where $n \in \{1, 2, 3, 4\}$; $\mathbf{e}^n$ is the deep representation of embedding interaction learned by the n-th layer in the CNN, ϕ_l^n is the l-th layer in the convolutional network n, $\mathbf{W}_l^j$ denotes the weight matrix, $\mathbf{b}_l^j$ denotes the bias, σ_l^n denotes the activation function, and $\mathbf{z}_0$ denotes the concatenation for pairwise correlations of user and item collective embedding.

4.4 Hidden Layer

The output of the pairwise interaction layer feeds into CNN in the hidden layer. The CNN is responsible for learning non-linear interactions from the interaction layer. We define the convolutional layer as $\mathbf{c} = F_\Theta(\mathbf{I})$, where F_Θ is the model of hidden layers with parameters $\mathbf{\Theta}$, and $\mathbf{c}$ is the feature maps vector used to predict the final score. We describe the details about the CNN in section 4.7.

4.5 Fusion Layer

The fusion layer is above the hidden layer, being responsible for combining four latent feature vectors into a single one as specified in Equation E.15.

$$\begin{aligned} \mathbf{c}_f &= \delta_f(\mathbf{W}_f \mathbf{z}_f + \mathbf{b}_f), \\ \mathbf{z}_f &= \mathbf{c}^1 \oplus \mathbf{c}^2 \oplus \mathbf{c}^3 \oplus \mathbf{c}^4 \end{aligned} \tag{E.15}$$

where $\mathbf{W}_f$ is the weight matrix, $\mathbf{b}_f$ is the bias vector, and δ_f is the activation function. $\mathbf{z}_f$ is the concatenation of four latent interaction representations.

4.6 Prediction Layer

The prediction layer uses the output vector $\mathbf{z}_f$ from the fusion layer. The prediction score is calculated considering the recommender system as a classification problem as defined by Rendle *et al.* [3]. First, let $\hat{r}_{vit} = \mathbf{w}^T \mathbf{z}_f$, where vector $\mathbf{w}$ denotes the weights for the user-item-time interactions in $\mathbf{z}_f$. Then, CoNCARS calculate the prediction score for user $\mathbf{v}$ and item $\mathbf{j}$ in time $\mathbf{t}$, resulting in a tuple $(\mathbf{v}, \mathbf{j}, \mathbf{t})$. A positive result of the tuple $(\mathbf{v}, \mathbf{k}, \mathbf{t})$ denotes that $\hat{r}_{vjt}$ should be larger than $\hat{r}_{vkt}$ to have the correct label of $+1$. On the other hand, a negative result has a label of 0.

4.7 Convolutional Layer

CNN architecture consists of complex concepts such as stride and padding as explained by Goodfellow *et al.* [16]. In this paper, we only focus on describing the most important settings concerning our network. The embedding size of the input layer CNN is 64×64. The channel has 6 hidden layers with 32 feature maps. The feature maps $\mathbf{c}$ in the hidden layer l is represented as a $2D$ matrix of the interaction layer $\mathbf{I}^{le}$. The stride is set as $[1, 2, 2, 1]$ which represents the *example*, *height*, *width*, and *channel*, respectively. Considering these settings, the size of $\mathbf{I}^{le}$ is half of its previous layer $l - 1$. We denote the feature maps of the first layer as a $3D$ tensor ε^l. The tensor in CNN allows CoNCARS to capture specific temporal patterns considering the common latent space. We define ε^l as:

$$\begin{aligned} \varepsilon^l &= [\mathbf{e}^l_{v,j,d}]_{32\times 32\times 32}, \quad \text{where} \\ \mathbf{e}^l_{v,j,d} &= ReLU(b_1 + \sum_{a=0}^{c-1} \sum_{b=0}^{c-1} e_{2v+a,2j+b} \cdot t^1_{1-a,1-b,d}), \end{aligned} \tag{E.16}$$

where b_1 is the bias for the first layer, and $T^1 = [t^1]2 \times 2 \times 32$ is a $3D$ tensor is the convolutional filter used to generate the learning representation in the first layer. The activation function is set as *ReLU* to convert negative values to zero. The same operation is applied in the following layers. However, now there is a $3D$ tensor ε^l as input to the next layer $l + 1$ as formally described as:

$$\begin{aligned}\varepsilon^{l+1} &= [\mathbf{e}^{l+1}_{v,j,g}]_{s\times s\times 32}, \quad \text{where} \quad 1 \le l \le 5, \quad s = \frac{64}{2^{l+1}} \\ \mathbf{e}^{l+1}_{v,j,g} &= ReLU(b_{l+1} + \sum_{a=0}^{c-1}\sum_{b=0}^{c-1} e_{2v+a,2j+b}.t^{l+1}_{1-a,1-b,g}),\end{aligned} \tag{E.17}$$

where b_{l+1} is the bias, while $[t^{l+1}_{a,b,d,g}]_{2\times 2\times 32\times 32}$ is the $4D$ convolutional filter for the following layer $l+1$. The output generated by the last layer is a tensor of dimension $1 \times 1 \times 32$ used by the prediction layer to calculate the final score with a weight vector $\mathbf{w}$.

4.8 Learning Algorithm

Element-wise squared loss has been widely used for latent factor recommendation models. However, it is not optimal for the ranking task. For a better optimization regarding top-N task, CoNCARS applies Bayesian Personalized Ranking (BPR) [3] as the pairwise optimization method.

During the representation learning CoNCARS uses mini-batch gradient descent to calculate the gradient with a small batch of samples, and formulate the training set as $\mathbf{M}$ in Equations E.18 and E.19.

$$\mathbf{M}_p = (p, q, q')|p \in \mathbf{P} \wedge q \in \mathbf{Q}^+_p \wedge q' \in \mathbf{Q}\backslash\mathbf{Q}^+_p \tag{E.18}$$

$$\mathbf{M}_t = (t, q, q')|t \in \mathbf{T} \wedge q \in \mathbf{Q}^+_t \wedge q' \in \mathbf{Q}\backslash\mathbf{Q}^+_t, \tag{E.19}$$

where $\mathbf{p}$ denotes the users latent factors, $\mathbf{t}$ represents the time, $\mathbf{q}$ denotes the positive feedback, and $\mathbf{q}'$ represents the non-interacted items. The objective function is defined as in Equation E.20.

$$\begin{aligned}\mathbf{L}_{BPR}(\mathbf{M}) = &\sum_{(p,x,x')\in\mathbf{M}_{px}} \ln\sigma(\hat{\mathbf{I}}_{p_u}) \\ &+\lambda_1 \sum_{(p,q,q')\in\mathbf{M}_{pq}} \ln\sigma(\hat{\mathbf{I}}_{q_i}) \\ &+\lambda_2 \sum_{(y,x,x')\in\mathbf{M}_{ut}} \ln\sigma(\hat{\mathbf{I}}_{y_u}) \\ &+\lambda_3 \sum_{(y,q,q')\in\mathbf{M}_{ui}} \ln\sigma(\hat{\mathbf{I}}_{x_i}) \\ &-\lambda_\Theta ||\mathbf{\Theta}||^2_F,\end{aligned} \tag{E.20}$$

where σ is the sigmoid function; $\mathbf{\Theta} = \{\mathbf{P}, \mathbf{Q}, \mathbf{T}\}$ and $\lambda_\Theta = \{\lambda_4, \lambda_5, \lambda_6\}$. The gradient of the objective function is given by Equation E.20 and maximized based on the first order derivatives as formulated in Equation E.21.

Algorithm 2 Mini-batch gradient descent method

```
Input  : user vector u, item vector i user-time vector ut, item-time vector it,
         regularization coefficients ˘$_\Theta$, batch size b, learning rate ȷ, number
         of epochs epoch_max, and convergence criteria.
Output: top-n prediction given by the prediction score $\hat{r}$.
initialize $\Theta$ randomly
 epoch = 0
 while not converged && epoch < epoch_max do
     epoch+=1
      shuffle all observed interaction
      split all item-user interactions into b-size mini-batches
      get the mini-batch in a sequential way
      foreach mini-batch do
          foreach record in current batch do
              select 5 non-observed items $\mathbf{q}'$ randomly from
                $\mathbf{Q}\backslash(\mathbf{Q}_p^+ \cup \mathbf{Q}_r^+)$
                add the negative samples to the current batch
          end
          calculate the gradient according to Equation E.21 for current batch
            $\boldsymbol{\Theta} = \boldsymbol{\Theta} + \jmath \nabla_\Theta \mathbf{L}_{BPR}(\mathbf{M})$
     end
     concatenate the output according to Equation E.15
     calculate $\hat{r}$ and predict the top-N items
end
return the top-N items
```

$$\begin{aligned}\nabla_\Theta \mathbf{L}_{BPR}(\mathbf{M}) = {} & \sigma(-\hat{\mathbf{I}}_{p_u}\frac{\partial \hat{\mathbf{I}}_{p_u}}{\partial \boldsymbol{\Theta}}) \\ & +\lambda_1\sigma(-\hat{\mathbf{I}}_{q_i}\frac{\partial \hat{\mathbf{I}}_{q_i}}{\partial \boldsymbol{\Theta}}) \\ & +\lambda_2\sigma(-\hat{\mathbf{I}}_{y_u}\frac{\partial \hat{\mathbf{I}}_{y_u}}{\partial \boldsymbol{\Theta}}) \\ & +\lambda_3\sigma(-\hat{\mathbf{I}}_{x_i}\frac{\partial \hat{\mathbf{I}}_{x_i}}{\partial \boldsymbol{\Theta}}) \\ & -\lambda_\Theta||\boldsymbol{\Theta}||_F^2,\end{aligned} \tag{E.21}$$

CoNCARS is trained using a mini-batch as shown in Algorithm 2. First, we randomly initialized the parameters, then for each iteration, we compute the gradients according to Equation (E.21) with a batch of **b** positive samples and 5 **b** negative samples to construct 5 **b** preference pairs, and update the parameters. Finally, CoNCARS predicts the top-N items.

4.9 Model Training

We train CoNCARS model based on the BPR objective [3] with Adagrad optimizer [17]. Prior computing the Algorithm 2, we pre-trained the embedding layer of CoNCARS using MF to avoid local optimal solutions, since the objective function is non-convex. After obtaining the parameters $\mathbf{\Theta}$, the first epoch of CoNCARS is trained using $L2$-regularization. Once the network is trained CoNCARS ranks the personalized list of items **i** for a user **u** in time **t** based on the value of $\hat{r}_{uit} = \mathbf{w}^T \mathbf{z}_f$ and $\hat{r}_{ujt} = \mathbf{w}^T \mathbf{z}_f$ on the set of items.

5 Empirical Evaluation

The empirical evaluation aims to answer the following research questions:

RQ1 Does the proposed method CoNCARS outperform the state-of-art methods for item recommendations?

RQ2 How do the number of feature maps **c** and λ affect the performance of CoNCARS?

5.1 Experimental Settings

Datasets. The experiments are performed on three datasets: MovieLens 10M[1], Yelp [2], and Pinterest [3]. They are public benchmark datasets for the recommender systems research community. We covert the datasets to implicit feedback following the Equation E.1, where 1 denotes a user interaction with an item, and 0 denotes otherwise. Table E.1 summarizes the statistics of each dataset. To address the sparseness of the datasets we apply the settings recommended by He *et al.* [2] and thereby only retrieving users with minimum 20 interactions.

Statistics	**Movielens**	**Yelp**	**Pinterest**
# of Users	6,040	25,815	55,187
# of Items	3,706	25,677	52,400
# of Interactions	1,000,209	730,791	1,5000,809
Sparcity	95.53%	99.89%	99.73%

Table E.1: Statistics of the Datasets

[1] https://grouplens.org/datasets/movielens/1m/

[2] https://github.com/hexiangnan/sigir16-eals

[3] https://sites.google.com/site/xueatalphabeta/academic-projects

- **Movielens 10M.** is a movie dataset which has been widely used to evaluate the accuracy of collaborative filtering methods. This version has more than one million user interactions, where each user rated at least 20 movies. The dataset contains explicit feedback which we converted to implicit feedback to investigate the performance of our model in this domain. [18].
- **Yelp.** is restaurant dataset obtained during the Yelp Challenge. In order to provide new items to a user, we merged the repetitive ratings at different timestamps to the earliest one as recommended by He *et al.* [2].
- **Pinterest.** is an implicit image dataset provided by Geng *et al.* [19] to evaluate content-based image recommendation. Each user-item interaction denotes whether the user has brought the image to her/his collection.

5.2 Evaluation Protocol

The evaluation of the item recommendation task using implicit feedback is performed using an adapted version of leave-one-out evaluation protocol [2, 20, 21]. The latest one user-item interaction is held-out considering it as a positive testing set, and the remaining interactions that the user did not rat before as training set. This strategy is computationally exhaustive. Therefore, to minimize the drawback, we follow the strategy applied by [18], where it randomly samples 100 items which are not interacted by the users. During the training step, CoNCARS ranks the list of all non-interacted items in the training set according to their prediction score. The top-N evaluation for item recommendation is based on the N number of items in the ranking list. The metrics used to evaluate the ranking list are Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [2]. HR measures whether the testing item is in the top-N list, considering the presence of an item as a hit. NDCG, on the other hand, measures the quality of the prediction based on the position of an item in the recommended list. In both metrics, high values denote better performance.

5.3 Baseline Methods

The following state-of-art methods are used as baselines to evaluate CoNCARS performance.

- BPR [3] optimizes the MF model of Equation E.2 by applying a pairwise loss function to learn from implicit feedback. We applied a fixed learning rate, which had the best performance in our experiments.

- CAMF [8] extends MF by using contextual features. In our experiments, we use time as a context to represent users interactions.

- TF [22] identifies the correlations between the user, item, and time using a three dimensional latent space. We set the embedding size with the best performance in our experiments.

- CHNMF [4] decomposes the user-item interaction into a common latent space combining users interactions, item's attributes, and contextual information. In our experiment, we consider only the time as the contextual feature and use the embedding size that had the best performance.

- Neural Matrix Factorization (NeuMF) [2] pre-trains multi-layer perceptron (MLP) network and MF separately, and then ensembles those two models to predict the user's preferences. We set the embedding as 16 for each layer, since larger values caused overfitting and dropped the performance.

- Convolutional Matrix Factorization (ConvMF) [14] integrates convolutional neural network (CNN) into probabilistic matrix factorization (PMF). ConvMF aims to capture contextual information of documents and further enhances the rating prediction accuracy. In our experiments, ConvMF is trained to predict the top-N items for implicit feedback.

5.4 Parameters Settings

We developed the CoNCARS model in Python based on Tensorflow framework. We tune the hyperparameters by keeping one training interaction for each user in the validation set. We optimize the regularization terms separately for the embedding layer, convolutional layers, and output layer in the range of $[0.001, 0.01, ..., 100]$. We set the embedding size as 64 for all baseline models and apply the BPR loss using Adagrad optimization with a learning rate of $\eta = 0.05$ and batch size of 512 for a fair comparison. We set the number of layers of NeuMF method between one and three as recommended by He *et al.* [2]. We pre-train the embedding layers of the neural baseline models using BPR and tuned their $L2$-regularization with the best values when achieving the best performance.

5.5 Performance Comparison (RQ1)

Table E.2 summarizes the performance comparison for top-N recommendation in the datasets. The analysis considers $N = 10$ and $N = 20$ list of items as they are generally used to express the effectiveness of item recommendation.

	Movielens				**Yelp**				**Pinterest**			
	HR@N		**NDCG@N**		**HR@N**		**NDCG@N**		**HR@N**		**NDCG@N**	
	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20	N=10	N=20
CAMF	0.4816	0.4929	0.1999	0.2517	0.0535	0.1006	0.0514	0.0591	0.5738	0.5911	0.3337	0.3619
TF	0.5395	0.5548	0.3076	0.4189	0.0991	0.2061	0.0953	0.0982	0.6914	0.7028	0.4574	0.4777
CHNMF	0.5500	0.5711	0.3244	0.4365	0.1121	0.2150	0.1001	0.1013	0.7164	0.7502	0.4918	0.5009
BPR	0.5841	0.6573	0.3664	0.4395	0.1558	0.2607	0.1042	0.1379	0.7464	0.8025	0.5119	0.5371
NeuMF	0.6774	0.7300	0.4133	0.4470	0.1841	0.2967	0.1095	0.1538	0.7593	0.8770	0.5324	0.5520
ConvMF	0.6801	0.7558	0.4171	0.4494	0.1937	0.3005	0.1102	0.1547	0.7921	0.8995	0.5383	0.5636
CoNCARS	**0.6974**	**0.8422**	**0.4235**	**0.4596**	**0.2442**	**0.3751**	**0.1220**	**0.1588**	**0.8801**	**0.9691**	**0.5588**	**0.5749**

Table E.2: Top-N recommendation performance at $N = 10$ and $N = 20$. The bold font indicates the best results.

CoNCARS outperforms ConvMF by 9% because it applies conventional element-wise product rather than the pairwise product to capture the user, item, and time correlations. Furthermore, ConvMF does not use temporal features to identify the user′s preferences according to different timestamps.

CoNCARS presents a relative improvement of 10, 25% when compared to NeuMF. The best performance achieved by CoNCARS relies on three factors the use of time as a feature, use of pairwise interaction layer and CNN to learn the non-linear features. Time has proved to be an essential feature to model the user-item interactions, due to providing a more accurate sequence of items in the top-N list. The pairwise product among the user, item, and time embeddings captures better correlations among the embedded representations than the element-wise product.

NeuMF applies MLP to learn non-linear interactions between the user and item, while CoNCARS uses CNN for this purpose. MLP has a good representation ability as described by Hornik *et al.* [23], however it uses a large number of parameters. For example, assuming the embedding size as 32 and 1-layer MLP would have around 4 million parameters, while our 6-layer CNN has around 20 thousand parameters. The number of parameters required by MLP may drop its performance to converge to the optimal solution when compared with methods using CNN.

Despite CoNCARS and BPR applying the same loss function, we observe a relative improvement of 24% of the first one compared to the last. The use of a collective embedding layer together with a CNN layer by CoNCARS increased its accuracy for top-N recommendation.

By comparing CoNCARS with CHNMF, we observe an average relative improvement of 37, 8%. The strategy of applying pairwise product by CoNCARS rather than point-wise squared loss as used by CHNMF, allowed CoNCARS to present a better learning process of user and item correlations over time. Furthermore, using CNN to identify non-linear correlations from the user, item, and time embeddings.

TF demonstrates a good representation learning for the user, item, and time embeddings. However, it fails to capture temporal dynamics due to the limitation of relying only on the current time-slot. Moreover, TF is not able

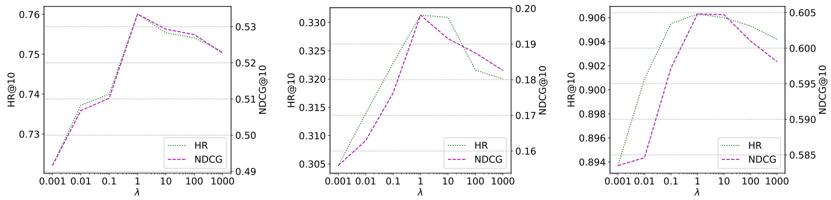


Fig. E.3: Performance of CoNCARS regarding to hyper-parameter λ on Movielens (left), Yelp (center), and Pinterest (right).

to identify non-linear correlations. The relative improvement observed gives CoNCARS a better performance of 43%.

CoNCARS has an average relative improvement of $65,5\%$ in comparison to CAMF. CAMF's poor performance may be caused by its use of time as a contextual feature in the conventional MF method, resulting in the inability to capture temporal patterns. Using conventional MF, CAMF is not able to capture the non-linear interactions as the neural models.

Among the baselines, ConvMF performs the best, proving the advantage of CNN to learn non-linear correlations between user-item interactions. CAMF had the weakest performance as it does not use multi-dimensional representation during the learning step. CHNMF and TF represent the correlations among user and items over time in a higher-dimensional representation. However, they do not optimize their objective function applying pairwise correlation, causing a weaker performance than ConvMF.

5.6 Hyper-parameters Analysis (RQ2)

CoNCARS has two hyper-parameters that can affect its effectiveness. λ and **c**. λ defines the regularization term, and **c** denotes the number of feature maps per convolutional layer. We analyze the results for different values of λ and **c** in three datasets: Movielens, Yelp, and Pinterest.

The results regarding the hyper-parameter λ are shown in Figure E.3 considering different values of λ. The curve observed in Figure E.3 leads to interpreting that values of λ smaller than 1 demonstrate gradual improvements in the performance when we analyze the convergence curve for the Movielens, Yelp, and Pinterest datasets. However, values larger than 1 keep the performance stable until a value of $1,000$.

Figure E.4 illustrates the top-N effectiveness for different values of hyper-parameter **c**. For all values of **c**, CoNCARS presents a gradual increase in the learning curve and finally reaches the convergence with a slight difference. The slight difference leads us to the conclusion that different values of **c** achieve similar performance, making CoNCARS suitable for real-world usage.

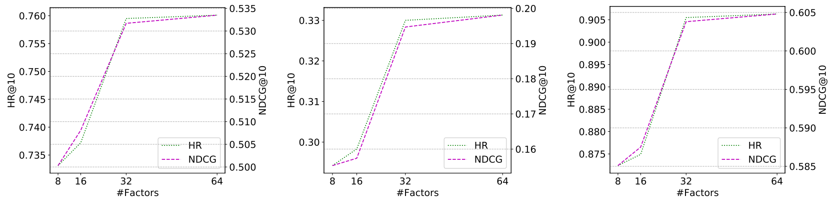


Fig. E.4: Performance of CoNCARS regarding the number of factors on Movielens (left), Yelp (center), and Pinterest (right).

6 Conclusion

We proposed a collective embedding for neural context-aware recommender systems for item recommendations using implicit feedback. We explain how CoNCARS uses temporal information in the feature embedding and why it is an important feature for the recommendation task. Furthermore, the paper described how the joint embedding and CNN improve the learning representation of the user and item interaction over time. Finally, we present the experimental results and CoNCARS performance regarding the hyper-parameters variance.

CoNCARS has proved to be useful for top-N item recommendations considering implicit feedback. Moreover, pre-training the model using MF with temporal information has presented good performance when used as input data to the CNN. Learning deep representations for pairwise interactions among user and item embedding over time improved the accuracy for predicting the user's preference score, as observed in the results of our experiments in the three benchmark datasets.

In the future, we will investigate other features that may influence in the prediction of the user's preferences, such as item's reviews [24] as richer contextual information due to a broader description given by users and user's viewpoints [25]. Another research direction is to model user's social relations and influence [26] since similar users tend to help in the decision-making of each other. Moreover, we would like to apply attention mechanism [27] to improve the recommendations based on user and item similarities.

7 Acknowledgments

The authors wish to acknowledge the financial support and the fellow scholarship given to this research from the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0).

References

[1] X. He, Z. He, X. Du, and T.-S. Chua, “Adversarial personalized ranking for recommendation,” in *SIGIR*, 2018, pp. 355–364.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW ’17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI ’09. Arlington, Virginia, United States: AUAI Press, 2009, pp. 452–461. [Online]. Available: http://dl.acm.org/citation.cfm?id=1795114.1795167

[4] F. Costa and P. Dolog, “Hybrid learning model with barzilai-borwein optimization for context-aware recommendations,” in *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida USA., May 21-23 2018.*, 2018, pp. 456–461.

[5] P. Bhargava, T. Phan, J. Zhou, and J. Lee, “Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data,” in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW ’15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 130–140. [Online]. Available: https://doi.org/10.1145/2736277.2741077

[6] P. G. Campos, F. Díez, and I. Cantador, “Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols.” *User Model. User-Adapt. Interact.*, vol. 24, no. 1-2, pp. 67–119, 2014.

[7] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, “Temporal collaborative filtering with bayesian probabilistic tensor factorization,” in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 211–222.

[8] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, “Incarmusic: Context-aware music recommendations in a car,” in *E-Commerce and Web Technologies*. Springer, 2011, pp. 89–100.

[9] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[10] X. He, M.-Y. Kan, P. Xie, and X. Chen, "Comment-based multi-view clustering of web 2.0 items," in *Proceedings of the 23th International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: ACM, 2014, pp. 771–782. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567975

[11] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proc. of Intl. Conf. on Data Mining*. SIAM, 2013, pp. 252–260.

[12] M. Saveski and A. Mantrach, "Item cold-start recommendations: Learning local collective embeddings," in *Proceedings of the 8th ACM Conference on Recommender Systems*, ser. RecSys '14. New York, NY, USA: ACM, 2014, pp. 89–96. [Online]. Available: http://doi.acm.org/10.1145/2645710.2645751

[13] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ser. WSDM '16. New York, NY, USA: ACM, 2016, pp. 153–162.

[14] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16. New York, NY, USA: ACM, 2016, pp. 233–240.

[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

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

[17] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, pp. 2121–2159, 2011.

[18] Y. 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*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401944

[19] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *Proceedings of the*

2015 IEEE International Conference on Computer Vision (ICCV), ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 4274–4282.

[20] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 3203–3209. [Online]. Available: https://doi.org/10.24963/ijcai.2017/447

[21] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, "Delf: A dual-embedding based deep latent factor model for recommendation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 3329–3335. [Online]. Available: https://doi.org/10.24963/ijcai.2018/462

[22] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 79–86. [Online]. Available: http://doi.acm.org/10.1145/1864708.1864727

[23] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, Mar. 1991.

[24] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1661–1670.

[25] Z. Ren, S. Liang, P. Li, S. Wang, and M. de Rijke, "Social collaborative viewpoint regression with explainable recommendations," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '17. New York, NY, USA: ACM, 2017, pp. 485–494.

[26] M. A. Saleem, F. S. da Costa, P. Dolog, P. Karras, T. Calders, and T. B. Pedersen, "Predicting visitors using location-based social networks," in *MDM*, 2018, pp. 245–250.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008.

Paper F

Convolutional Adversarial Latent Factor Model for Recommender System

Felipe Costa and Peter Dolog

The paper was accepted and will be published in the
Proceedings of the Thirty-Second International Florida Artificial Intelligence Research Society Conference, FLAIRS 2019, Sarasota, Florida USA, 2019.

Abstract

The high accuracy of Top-N recommendation task is a challenge in the systems with mainly implicit user feedback considered. Adversarial training has presented successful results in identifying real data distributions in various domains (e.g., image processing). Nonetheless, adversarial training applied to the recommendation is still challenged especially by interpretation of negative implicit feedback causing it to converge slowly as well as affecting its convergence stability. The researchers [1, 2] denotes the misinterpretation to high sparsity of the implicit feedback and discrete values characteristic from items recommendation. To face these challenges, we propose a novel model named convolutional adversarial latent factor model (CALF), which uses adversarial training in generative and discriminative models for implicit feedback recommendations. We assume that users prefer observed items over generated items and then apply the pairwise product to model the user-item interactions. Additionally, the hidden features become input data of our convolutional neural network (CNN) to learn correlations among embedding dimensions. Finally, Rao-Blackwellized sampling is adopted to deal with the discrete values optimizing CALF and stabilizing the training step. We conducted extensive experiments on three different benchmark datasets, where our proposed model demonstrates its efficiency for item recommendation.

The layout has been revised.

1 Introduction

The internet has been facing an information overload due to a large amount of shared data on the Web. Recommender systems have been studied to help to overcome the information overload problem, aiming to predict user's preferences based on her/his history or popular items. Collaborative filtering (CF) has been the most commonly used method [3, 4]. Among the CF techniques, matrix factorization (MF) has become the most popular [5, 6] due to high accuracy in modeling the interaction between users and items such as browsing, rating, and clicking in latent space. Lately, implicit feedback has been extensively explored due to its practicality and accessibility in online services, turning the goal of recommender systems from rating predictions to learning to rank. The recommendation model aims to predict a personalized ranking over a set of items for each user based on the similarities among the users and items. Nevertheless, unobserved items from implicit feedback lead us to misinterpret negative values because they may be unseen items or items whose a user did not like.

To handle the research challenges mentioned above, we explore adversarial training to model users preferences from implicit feedback. Following the concept given by [7] Generative Adversarial Networks (GAN) have two components: a generative model trying to generate real samples and a discriminative model discriminating whether the samples are real or not. The idea is to train the model to defend against an adversary, such as the fake samples. Adversarial training has gained success in image processing and natural language processing, however in recommender systems it faces two issues: highly sparse data and discrete item values. The sparse data may cause gradient vanishing or update instability, and the discrete values do not allow the adversarial training to directly optimizes using the gradient descent. Recently, [8] proposed Information Retrieval GAN (IRGAN) which applies adversarial training in the information retrieval field. IRGAN uses policy gradient strategy to obtain the model parameters. However, the variance of the estimated gradients increases linearly according to the number of items, making this solution impractical in recommender systems, since a large amount of items may increase the vulnerability of adversarial training.

In order to solve the research challenges mentioned above, we propose a new adversarial training model for implicit recommender systems named CALF. Considering the adversarial training as a battle, the generative model aims to identify the user preferences by fighting with the discriminative model. The discriminative model aims to estimate the distribution distance between the generative model and the user preferences, while the generative model aims to minimize the estimated distance by capture the actual distribution. Assuming the user prefers observed items over the generated items and the adversarial training as a battle, the generative and the discriminative models are opponents alternately optimizing the pairwise loss function. The goal is to improve the discriminator's judgment by minimizing the pairwise objective function, while the generator tries to generate user preferred items. The adversarial training process helps to handle the negative samples and avoiding to design specific sampler as in the policy gradient method. Moreover, we replace the non-differentiable item sampling by a differentiable item generating procedure using Rao-Blackwellized sampling allowing the convolutional adversarial latent factor model (CALF) to update the gradients derived from the discriminator into the generator smoothly, allowing CALF to be trained by the standard gradient descent method rather than policy gradient.

The paper presents the following contributions:

- A new model named CALF to improve the prediction based on user's preferences;
- An adversarial training model for a better learning correlation between the embedding dimensions and accelerating the convergence;
- A differentiable sampling method to deal with the discrete values allowing CALF to optimizes with gradient descent;
- Empirical evaluation in three benchmark datasets demonstrates the effectiveness of the CALF model.

2 Related Works

Extensive work has been done in recommendations using explicit feedback. However, many online services rely on implicit feedback, where the main task from the recommendation system perspective is to provide a personalized list of items to each user rather than to predict the user ratings. Researchers from the recommender systems field have been investigating neural network techniques applied to collaborative filtering due to their ability to learn feature representations.

Restricted Boltzmann machines have proven better performance than traditional MF to perform CF [9], where [10] incorporate correlation between users and items to increase their efficiency. Other studies have focused on sequential recommendations such as music [11], where their goal is to learn the content features of items. Furthermore, some researchers incorporate multiple features of users and items into their recommender model, for example, WideDeep from Google [12] and D-Attn from [13]. However, in this paper, we focus on adversarial training for collaborative filtering.

[14] introduced NCF to model user-item interaction function with implicit feedback combining a shallow MF-based neural network with a multi-layer perceptron. Recently, an extended version named CNCF [1] uses an interaction map layer applying the outer product to model pairwise correlations between embedding dimensions. Furthermore, the embedded vectors are used as input for the CNN to learn the user-item interactions. Despite the effectiveness of NCF and CNCF, the neural collaborative filtering models have neglected adversarial perturbations causing vulnerabilities in their performance. Therefore, researchers proposed adversarial training for collaborative filtering, such as Adversarial Matrix Factorization (AMF) [15], IRGAN [8]), and CFGAN [2]. AMF applies adversarial personalized ranking (APR) on the MF method. IRGAN uses adversarial training into information retrieval field through element-wise product and defines the objective function via a probability-based method. CFGAN explores vector-wise adversarial training to solve the problem of discrete items. However, in both IRGAN and CFGAN the discriminative model performs as a binary classifier whose predicted values represent the probability that a user liked an item. In contrast, CALF is a pairwise method applying a generative and discriminative model based on CNN learning using adversarial training to learn user-item interactions.

3 Problem Formulation

The research problem investigated in the paper is defined as follows: *How to improve Top-N recommendation task using convolutional adversarial latent factor model?*

Consider a set of users $U = \{u_1, ..., u_M\}$ and a set of items $I = \{i_1, ..., i_N\}$. Let $Y = \{y_{vj} \in \mathbb{R}^{M \times N} | 1 \leq y_{vj} \leq 5\}$ denotes the rating matrix, where y_{vj} is the rating of user v on item j, and we label as *unk* if it is unknown. Then, model the matrix Y with implicit feedback as Eq. F.1.

$$Y_{vj} = \begin{cases} 0, & \text{if } y_{vj} = unk \\ 1, & \text{otherwise} \end{cases} \tag{F.1}$$

The latent factor models define the recommender systems as the problem of predicting the preference score of each unobserved entry in Y to further rank the list of items. We generate the scores as defined in Eq. F.2.

$$\hat{Y}_{vj} = F(u_v, i_j | \Theta) \tag{F.2}$$

where $\hat{Y}_{vj}$ is the predicted score of interaction Y_{vj} between user u_v and item i_j, Θ is the model parameters, and F is the function which estimates the predicted scores based on Θ. The function F leads to find the optimal list of items for an individual user according to users preferences.

Traditionally, MF methods define the function F based on element-wise product of p_v and q_j to predict $\hat{Y}_{vj}$ as demonstrated by [5], where p_v and q_j defines the hidden latent factors of u_v and i_j, respectively.

$$\hat{Y}_{vj} = F(u_v, i_j|\Theta) = \mathbf{p}_v^T \mathbf{q}_j \tag{F.3}$$

We apply the pairwise product to calculate the interactions between users and items. The advantage in comparison with the element-wise product is a better representation model for non-linear interactions between users and items through a deep learning architecture. The loss function of the pairwise method follows the strategy given by [16] where the difference between the items' ranking scores is given by:

$$L_{vjk} = \ln \sigma(\hat{\mathbf{y}}_{vj} - \hat{\mathbf{y}}_{vk}), \tag{F.4}$$

where $\hat{\mathbf{y}}$ is a ranking score, and a small loss represents high confidence that user v prefers item j over item k.

We use the following notations in further sections: u denotes a user and i denotes an item; v and j are indexes used to represent u and i, respectively. Y defines the user-item rating matrix mapped using Eq. F.1, where I^+ is the observed interactions and I^- unobserved interactions. Finally, (v, j) denotes the $-th$ element from the matrix Y.

4 Proposed Model

CALF is illustrated in Figure F.1, where the design of the prediction model defined in Eq. F.2 can be observed. In the following section, we describe the detailed architecture of the CALF model.

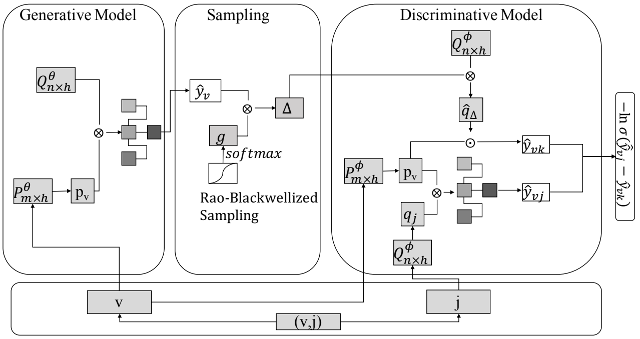


Fig. F.1: Convolutional Adversarial Latent Factor Model

4.1 CALF Architecture

CALF has a generator $\mathbf{g}_\theta$ and a discriminator $\mathbf{d}_\phi$, where θ and ϕ are the parameters for the generative and discriminative models, respectively. Furthermore, $\mathbf{s}_\theta$ denotes the generator distributions and $\mathbf{s}_{real}$ denotes the user's true preferences. Specifically, the generator $\mathbf{g}_\theta$ tries to generate personalized items for each user through minimizing the distance between $\mathbf{s}_\theta$ and $\mathbf{s}_{real}$. While the discriminator $\mathbf{d}_\phi$ discriminates whether a user u prefers item j over k.

The convolutional layer in our generator and discriminator is inspired by [1]. The CNN is responsible for processing the useful signal from the pairwise user-item interactions. The embedding size of the input layer CNN is 64×64. The channel has 6 hidden layers where each of them has 32 feature maps. A feature map $\mathbf{y}$ in the hidden layer l is represented as a $2D$ matrix of the interaction layer $\mathbf{S}^{ly}$. The stride is set as $[1,2,2,1]$ which represents the *example*, *height*, *width*, and *channel*, respectively. The padding is defined as $SAME$. Considering these settings, the size of $\mathbf{S}^{lc}$ is half of its previous layer $l-1$. Let $\mathbf{y} = F_{(\mathbf{g}_\theta|\mathbf{d}_\phi)}$, where $F_{(\mathbf{g}_\theta|\mathbf{d}_\phi)}$ is the model function with the parameters θ and ϕ. $\mathbf{y}$ is the mapping feature vector used to predict the final score.

Assuming that the user prefers the observed items over the generated items, the training step in the discriminator is a tuple (u, j, k), where user $u \in U$, item $j \in I_u^+$ and item k is sampled from $\mathbf{s}_\theta(k|u)$. The discriminator objective function is defined as:

$$J(\mathbf{g}_\theta, \mathbf{s}_\phi) = \max_\theta \min_\phi \sum_{u=1}^{m} \mathbb{E}_{j\sim\mathbf{s}_{real}(j|u)\&\&k\sim\mathbf{s}_\theta(k|u)} \ln \sigma(\hat{\mathbf{y}}_{vj} - \hat{\mathbf{y}}_{vk}), \tag{F.5}$$

where $\ln \sigma(\cdot)$ is the pairwise loss function. CALF adopts the logistic loss function proposed by [17], where the discriminator approximates the distance between $\mathbf{s}_\theta$ and $\mathbf{s}_{real}$. Note, while the discriminator in CALF minimizes the objective function and the generator maximizes the objective function, in the original GAN the loss function behaves oppositely.

4.2 Sampling Strategy

Due to the discrete nature of item sampling, the gradients derived from the objective function can not directly feed the generator. To solve this problem, IRGAN applies the policy gradient (reinforcement learning) to estimate the generator's gradients. However, the policy gradient presents two significant drawbacks: unstable training and slow convergence. To avoid these issues [18] propose to relax the discrete items. Rao-Blackwellization [19] proposes to reduce the variance of stochastic gradient estimators. In this paper, Rao-

Blackwellization is adopted to solve the issues with the discrete items; then, we optimize CALF using gradient descent.

For user u, we denote $\hat{\mathbf{v}}_u \in \mathbb{R}^n$ as the the vector of item ranking scores and $\mathbf{g}_\Delta$ as the adversarial perturbations vector whose elements are randomly resulted from Rao-Blackwellization $(0, 1)$. The sampling is defined as:

$$\boldsymbol{\Delta} = \frac{1}{n} \sum_{i=1}^{n} \frac{2\hat{\mathbf{v}}_u + \mathbf{g}_{\boldsymbol{\Delta}}}{g_i + 1}, \tag{F.6}$$

where $\boldsymbol{\Delta}$ is the generated analogous one-hot item vector. To differentiate $\boldsymbol{\Delta}$ from the real items, we name it as a fake item. Each real item has a correspondent feature vector. However, it is not possible to define a feature vector for each fake item, because it can exist an infinite number of them. Therefore, we define a differentiable method to obtain the feature vector of each fake item as:

$$\hat{\mathbf{q}} = \boldsymbol{\Delta}\mathbf{Q}, \tag{F.7}$$

where $\boldsymbol{\Delta} \in \mathbb{R}^n$ is a fake item, $\mathbf{Q} \in \mathbb{R}^{n \times h}$ is the fake items embedding matrix and $\hat{\mathbf{q}} \in \mathbb{R}^h$ is the feature vector of item $\boldsymbol{\Delta}$. h is the number of hidden features. The described strategy proposes to overcome the discrete items challenge and facilitates the gradient information updates into the generator.

The parameters ϕ from the discriminator can now update via gradient descent aiming to minimize the objective function:

$$\phi \leftarrow \phi - lr \times \nabla_\phi \ln \sigma(\hat{\mathbf{y}}_\phi(v_j) - \hat{\mathbf{y}}_\phi(v_k)). \tag{F.8}$$

On the other hand, the parameters θ from the generator aim to maximize the objective function and optimizes using gradient ascent:

$$\theta \leftarrow \theta + lr \times \nabla_\theta \ln \sigma(\hat{\mathbf{y}}_\theta(v_j) - \hat{\mathbf{y}}_\theta(v_k)). \tag{F.9}$$

lr denotes the learning rate. The proposed algorithm is described in Algorithm 3.

5 Empirical Evaluation

We describe the experimental setup used to evaluate the CALF model performance explaining the datasets, evaluation metrics, baseline methods, and CALF settings. Moreover, we define the following research questions:

RQ1 Does the proposed model, CALF, outperform the state-of-art methods for item recommendations?

Algorithm 3 CALF Algorithm

```
Input : generator $\mathbf{g}_\theta$, discriminator $\mathbf{d}_\phi$, user-item interactions $Y$, learning
        rate $\jmath$, number of epochs epoch_max, and convergence criteria.
Output: top-n prediction from the prediction score $\hat{y}$.
initialize $\theta$ and $\phi$ randomly
epoch = 0
while not converged && epoch < epoch_max do
    epoch+=1
    shuffle all observed interaction
    foreach discriminator step do
        foreach observed feedback $(u, j)$ in current batch do
            compute the generator items ranking scores $\hat{\mathbf{v}}_u$ for user $u$
            generate a fake item $k$ from $g_\theta$ for user $u$ via Equation F.6
            get feature vector of fake item $k$ via Equation F.7
            compute the pairwise loss $\ln \sigma(\hat{y}_{vj} - \hat{y}_{vk})$
            update the discriminator parameters $\phi$ via Equation F.8
        end
    end
    foreach generator step do
        foreach observed feedback $(u, j)$ in current batch do
            compute the generator items ranking scores $\hat{\mathbf{v}}_u$ for user $u$
            generate a fake item $k$ from $g_\theta$ for user $u$ via Equation F.6
            get feature vector of fake item $k$ via Equation F.7
            compute the pairwise loss $\ln \sigma(\hat{y}_{vj} - \hat{y}_{vk})$
            update the discriminator parameters $\theta$ via Equation F.9
        end
    end
end
return the top-$N$ items
```

RQ2 Does the Rao-Blackwellization sampling strategy outperform the policy gradient method?

RQ3 Does CALF improve the training convergence?

5.1 Experimental Settings

Datasets. The experiments of the CALF model and baselines were conducted on three datasets: MovieLens 10M[1], Yelp [2], and Pinterest [3]. They

[1] https://grouplens.org/datasets/movielens/1m/

[2] https://github.com/hexiangnan/sigir16-eals

[3] https://sites.google.com/site/xueatalphabeta/academic-projects

are public benchmark datasets for recommender systems research community, and publicly available on their websites. The datasets were converted to implicit feedback following Eq. F.1, where 1 denotes a user interaction with an item and 0 otherwise. Table F.1 presents the statistics of the three datasets. We consider only users with minimum 20 interactions as recommended by [14] due to high sparsity in the datasets.

Statistics	**Movielens**	**Yelp**	**Pinterest**
# of Users	6,040	25,815	55,187
# of Items	3,706	25,677	52,400
# of Interactions	1,000,209	730,791	1,5000,809
Sparcity	95.53%	99.89%	99.73%

Table F.1: Statistics of the Datasets

Evaluation Metrics. To evaluate item recommendation using implicit feedback, the widely used leave-one-out evaluation protocol [1, 14] was applied. The latest user-item interaction is held-out as the testing set, and the remaining interactions as the training set. The strategy applied by [6] is adopted to minimize the time consumed during the evaluation of the top-N recommendation task. Then, randomly sample 100 items which are not interacted by the users. During the training process, the personalized ranking of all items which are not interacted by the user in the training set according to the prediction score. To evaluate the performance of item recommendation considering the top-N task, we truncate the ranking list at position N. The metrics used to evaluate the ranking list are Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [14]. HR measures whether the testing item is in the top-N list. NDCG considers not only the presence of an item in the top-N list and its position.

Baseline Methods. The following state-of-art methods are used as baselines to compare the effectiveness of the CALF method.

- AMF [15] applies adversarial personalized ranking (APR) on the shallow state-of-art MF method demonstrating good improvements in different datasets.
- CNCF [1] applies the outer-product pairwise model to learn the correlations between the embedding dimensions, and CNN in their hidden layer to learn the correlations in hierarchical steps;
- IRGAN [8] applies the element-wise method to the generative and discriminative models, where the discriminator is a binary classifier, and it

	Movielens				Yelp				Pinterest				
	HR@N		NDCG@N		HR@N		NDCG@N		HR@N		NDCG@N		RI
	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	N=5	N=10	
AMF	0.5331	0.7255	0.3517	0.4444	0.1176	0.2385	0.0950	0.1065	0.7098	0.8972	0.4946	0.5658	+31%
IRGAN	0.5400	0.7301	0.3744	0.4665	0.1321	0.2550	0.1035	0.1113	0.7200	0.9002	0.5111	0.5832	+25%
CNCF	0.6103	0.8041	0.4316	0.5011	0.1578	0.2686	0.1073	0.1200	0.7489	0.9026	0.5367	0.5881	+20%
CFGAN	0.6805	0.8352	0.4991	0.5640	0.1829	0.2889	0.1184	0.1459	0.7668	0.9053	0.5513	0.5928	+15%
CALF	**0.7124**	**0.8596**	**0.5121**	**0.6153**	**0.2037**	**0.3148**	**0.1364**	**0.1681**	**0.7811**	**0.9155**	**0.5742**	**0.6159**	-

Table F.2: Top-N recommendation performance at $N = 5$ and $N = 10$. The bold font indicates the best results. RI indicates the relative improvement of CALF over the corresponding baseline on average.

uses probability to obtain the objective function. Furthermore, IRGAN applies reinforcement learning to handle the discrete item problem.

- CFGAN [2] proposes a vector-wise adversarial training to deal with the discretization of items.

Modeling Settings. We implement the CALF model in Python based on Tensorflow framework. CALF achieves the best performance in our experiments with the hyper-parameters set as below:

- The embedding size of 64 and optimized the loss function using mini-batch Adagrad with a batch size of 512;
- The learning rate lr for both embedding and CNN is set as grid search from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$;
- The adversarial regularization term λ is set to 1 to ensure that the discriminator satisfies the Lipschitz constraint regarded by the logistic loss.

5.2 Performance Comparison (RQ1)

Table F.2 summarizes the results regarding the performance comparison for top-N recommendation in the datasets. The analysis was made considering $N = 5$ and $N = 10$ as they are generally used to express the effectiveness of item recommendation.

CALF has an average relative improvement of 31% when compared to AMF. AMF applies adversarial perturbations to the shallow MF method, what may cause poor performance in comparison to other methods using CNN. Indicating that CNN has a better learning curve when representing users and items embeddings considering the item recommendation task.

IRGAN presents a good overall performance. However, due to its use of the element-wise product, CALF outperforms it with a relative improvement of 25%.

CNCF proves the advantage of applying pairwise correlations and using CNNs to learn non-linear user-item correlations. However, CNCF is not

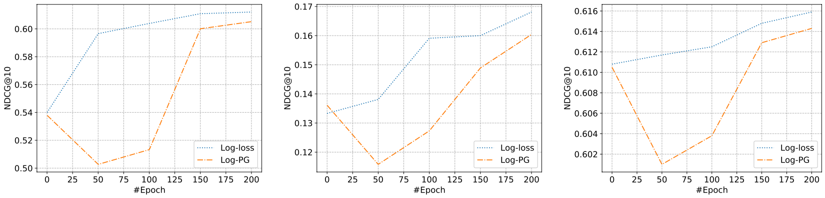


Fig. F.2: Differentiable sampling and policy gradient performance on Movielens (left), Yelp (center), and Pinterest (right).

able to refine the relevance between user and items because it does not apply adversarial training. CALF presents a relative improvement of 20% in comparison to CNCF.

CALF outperforms CFGAN with a relative improvement of 15% due to its use of pairwise training and sample strategy, facilitating the adversarial training. Furthermore, CALF applies CNN, demonstrating a higher accuracy when learning non-linear user and item embeddings.

The results present the effectiveness of CALF, achieving the best overall performance in all datasets.

5.3 Sampling Strategy Effectiveness(RQ2)

The sampling strategy for discrete values was evaluated considering CALF using logistic loss function with policy gradient. The hyper-parameters from logistic loss and policy gradient have the same values for a fair comparison.

As illustrated in Figure F.2 the learning curve of logistic loss with gradient descent is stable in comparison with policy gradient. Analyzing the Figure F.2, a peak is observed in the policy gradient loss when the adversarial training starts, but after some epochs, it drops, and, finally starts increasing again. On the other hand, the logistic loss using the sampling strategy for discrete values keeps stable during the training step. In other words, the policy gradient has slower convergence, which may be caused by the high variance of the policy gradient. Reducing the gradient variance improves the adversarial training.

5.4 Time Complexity Analysis (RQ3)

The complexity analysis of GAN models is $O(|Y|nh)$, where $|Y|$ denotes the number of user-item interactions. CALF has additional computations when compared, for example, with the policy gradient applied by IRGAN. However, the computational time in adversarial models relies on the iterations. Moreover, the generative and discriminative models of CALF alternates the

	Movielens					Yelp					Pinterest				
	CALF	**AMF**	**CNCF**	**IRGAN**	**CFGAN**	**CALF**	**AMF**	**CNCF**	**IRGAN**	**CFGAN**	**CALF**	**AMF**	**CNCF**	**IRGAN**	**CFGAN**
D	1 m	-	-	45 s	50 s	1.7 m	-	-	55 s	1.3 m	1.9 m	-	-	1.9 m	1.9 m
G	1.7 m	-	-	1.5 m	1.2 m	3.9 m	-	-	1.7 m	2.5 m	3.7 m	-	-	4.9 m	3.9
EC	50	60	50	60	60	50	100	90	120	100	70	90	80	70	70
TC	3 h	4 h	3.5 h	5 h	4.6 h	4.6 h	5.3 h	5 h	7 h	6.5 h	6.5 h	6.9	6.8 h	8 h	7 h

Table F.3: Convergence time. - denotes the methods without generative and discriminative models; D is the discriminative model; G is the generative model; EC denotes the epoch convergence; and TC denotes the time convergence

optimization in each step, while in the other GANs the training epoch of the discriminator spends double time than the generator.

Table F.3 presents the computational time spent by CALF in comparison with other models. Analyzing the Table F.3 CALF spends more time to train the generator and discriminator in each epoch, however, the convergence time is shorter compared to the other models. Therefore, considering the total training time, the sampling strategy adopted by CALF improves the computational time and the stability.

6 Conclusion

We proposed a convolutional adversarial latent factor model for items recommendations using implicit feedback, named CALF. A detailed description explains how CALF uses adversarial training for its recommendation. Furthermore, we presented the results of the conducted experiments.

CALF has proved to be useful for top-N items recommendations considering implicit feedback. Moreover, learning deep representations for pairwise interactions among user and item embeddings improved the accuracy for predicting the user's preference score, as observed in the results of our experiments in the three benchmark datasets.

In the future, we will conduct investigations regarding richer contexts such as social relations and user's reviews. Moreover, we would like to apply attention mechanisms to learn user and item similarities.

7 Acknowledgments

The authors wish to acknowledge the financial support and the fellow scholarship given to this research from the Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant# 206065/2014-0).

References

[1] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *IJCAI*, 2018, pp. 2227–2233.

[2] D.-K. Chae, J.-S. Kang, S.-W. Kim, and J.-T. Lee, "Cfgan: A generic collaborative filtering framework based on generative adversarial networks," in *CIKM*, 2018, pp. 137–146.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*, 2001, pp. 285–295.

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *TKDE*, pp. 734–749, 2005.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[6] Y. 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*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401944

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *CoRR*, 2014.

[8] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *SIGIR*, 2017, pp. 515–524.

[9] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *ICML*, 2007, pp. 791–798.

[10] K. Georgiev and P. Nakov, "A non-iid framework for collaborative filtering with restricted boltzmann machines," in *ICML*, 2013, pp. III–1148–III–1156.

[11] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, pp. 66:1–66:36, 2018.

[12] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *DLRS*, 2016, pp. 7–10.

[13] S. Seo, J. Huang, H. Yang, and Y. Liu, “Interpretable convolutional neural networks with dual local and global attention for review rating prediction,” in *Proceedings of the 11th ACM Conference on Recommender Systems*, ser. RecSys’17, 2017, pp. 297–305.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW ’17. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 173–182. [Online]. Available: https://doi.org/10.1145/3038912.3052569

[15] X. He, Z. He, X. Du, and T.-S. Chua, “Adversarial personalized ranking for recommendation,” in *SIGIR*, 2018, pp. 355–364.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, “Bpr: Bayesian personalized ranking from implicit feedback,” in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI ’09. Arlington, Virginia, United States: AUAI Press, 2009, pp. 452–461. [Online]. Available: http://dl.acm.org/citation.cfm?id=1795114.1795167

[17] S. Rendle and C. Freudenthaler, “Improving pairwise learning for item recommendation from implicit feedback,” in *WSDM*, 2014, pp. 273–282.

[18] C. J. Maddison, A. Mnih, and Y. W. Teh, “The concrete distribution: A continuous relaxation of discrete randomvariables,” *CoRR*, 2016.

[19] R. Liu, J. Regier, N. Tripuraneni, M. I. Jordan, and J. McAuliffe, “Rao-blackwellized stochastic gradients for discrete distributions,” *CoRR*, 2018.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-391-4

AALBORG UNIVERSITY PRESS