Neural Explainable Collective Non-negative Matrix Factorization for Recommender Systems

Felipe Costa and Peter Dolog
Aalborg Uiversity, Selma Lagerløfs Vej 300, 9220, Aalborg, Denmark

Keywords: Explainability, Recommender Systems, Matrix Factorization.

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.

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 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.

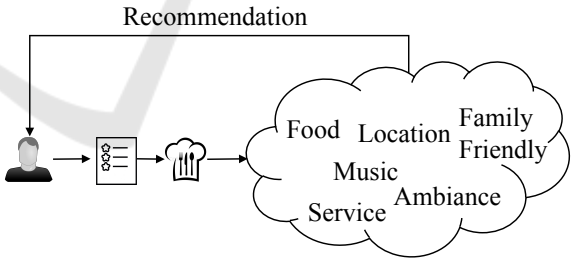

Figure 1: Example of Review-aware Recommendation.

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 prefer-

Costa, F. and Dolog, P.
Neural Explainable Collective Non-negative Matrix Factorization for Recommender Systems.
DOI: 10.5220/0006893700350045
In *Proceedings of the 14th International Conference on Web Information Systems and Technologies (WEBIST 2018)*, pages 35-45
ISBN: 978-989-758-324-7

ences. 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 (Zhang et al., 2014) 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 (He et al., 2015) 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, (Ren et al., 2017) 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 (Bengio et al., 1994). To solve this issue, long short-term memory (LSTM) have been introduced by (Hochreiter and Schmidhuber, 1997; El Hihi and Bengio, 1995) revealing good results in generating text upon different datasets (Karpathy et al., 2015) and machine translation (Sutskever et al., 2014).

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) (Cremonesi et al., 2010) and PageRank (Haveliwala, 2002), since they present reasonable results and simple implementation. Another widely applied method is matrix factorization (MF), for example SVD (Koren et al., 2009) and NMF (Lee and Seung, 2000). 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 (Liu et al., 2013; He et al., 2014). (Liu et al., 2013) proposed MultiNMF, using a connection between NMF and PLSA, and (He et al., 2014) 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 (Saveski and

Mantrach, 2014), 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 (Costa and Dolog, 2018), 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 (Zhang et al., 2014). 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. (Zhang et al., 2014) proposes the explicit factor model based on sentiment lexicon construction and (He et al., 2015) 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 (Costa et al., 2018).

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 1.

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

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.

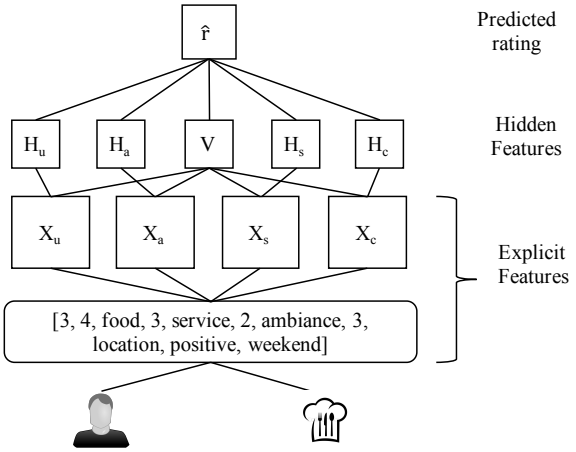

Figure 2: Collective Non-Negative Matrix Factorization.

Considering an online restaurant service as seen in Figure 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:

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

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.

4.1.1 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 (Saveski and Mantrach, 2014).

We model the local geometric structure through a nearest neighbour graph on a scatter of data points as seen in (Saveski and Mantrach, 2014). 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:

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

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:

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

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 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 5 before calculating the update rules.

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

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 6, after calculating the derivatives of $f(W)$, $f(H_u)$, $f(H_a)$, $f(H_c)$, and $f(H_s)$ from Equation 5.

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

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 7:

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

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 8 formally describes the statement as:

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

Applying ε_H in Equation 8 lead us to the following regularization term $\|P[H - \alpha\nabla f(H)] - H\|_F \leq \varepsilon_H$, where $\varepsilon_H = max(10^{-3}, \varepsilon)\|P[H - \alpha\nabla f(H)] - H\|_F$. We decrease the stopping tolerance by $\varepsilon = 0.1\varepsilon_H$, if the Borzilai-Borwein algorithm defined in (Costa and Dolog, 2018) solves Equation 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 4 defines $W^T W$ with dimensions $k \times k$ and $k \ll min\{m,n\}$. For a given $H_0 \geq 0$:

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

Following the definition of Equation 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 3. First, users and items embeddings are learned from *doc2vec* model (Le and Mikolov, 2014), 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.

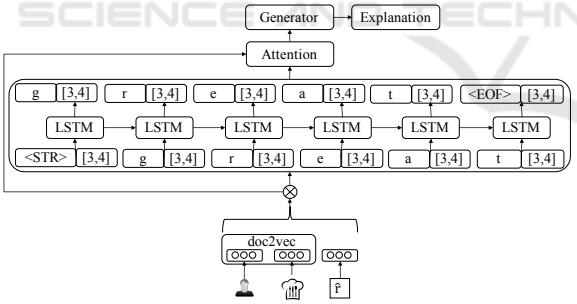

Figure 3: Natural Language Explainability model.

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. 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 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 10.

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

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 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 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.

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

Attention Layer. The attention layer adaptively learns soft alignments h_t between character dependencies c_t and $attention$ inputs. Equation 12 formally describes the character dependencies using attention layer $h_t^{attention}$ as explained by (Dong et al., 2017).

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

Generator Model. The generator model is built on character-level. To do so, we have to maximize a nonlinear *softmax* function to compute the conditional

probability p among the characters, as presented in Equation 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 13.

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

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 (Zhang et al., 2014; He et al., 2015). 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 1 summarizes the dataset information.

Table 1: Dataset Summarization.

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

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 14, where the metric gives higher score to the items at the top rank, and lower scores to the items at the low rank.

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

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:

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

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.

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

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**

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 (He et al., 2015), 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 hyperparameter to define the number of most cared features, which we define $k = 45$ as reported in (Zhang et al., 2014) 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 (He et al., 2015) 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.

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 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 2 we noticed PageR-

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

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 .

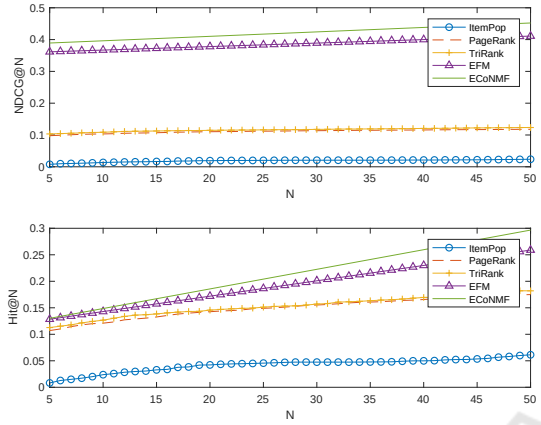

Figure 4: Empirical evaluation on Yelp dataset for N from 5 to 50.

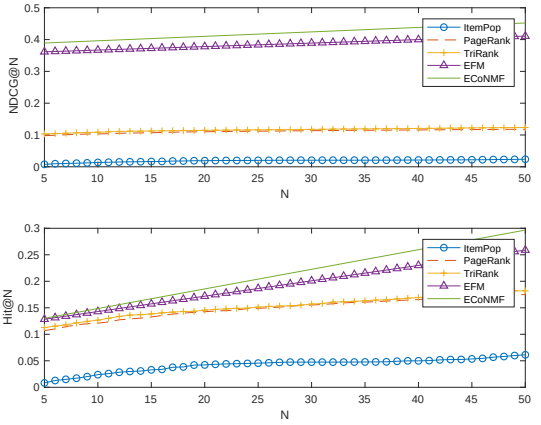

Figure 5: Empirical evaluation on Amazon dataset for N from 5 to 50.

ank 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 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 4 and 5 we observe NECoNMF presents higher accuracy performance for top-N recommendation when N varies 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 (Zhang et al., 2014), where they assess the quality of explanations throughout examples generated by the explainable model.

Table 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 regarding 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 (Zhang et al., 2014) explanations, we observe a better readability in Table 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.

ACKNOWLEDGEMENTS

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

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Costa, F. and Dolog, P. (2018). 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.*, pages 456–461.

Costa, F., Ouyang, S., Dolog, P., and Lawlor, A. (2018). Automatic generation of natural language explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, IUI'18, pages 57:1–57:2. ACM.

Cremonesi, P., Koren, Y., and Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA. ACM.

Dong, L., Huang, S., Wei, F., Lapata, M., Zhou, M., and XuT, K. (2017). Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, CECACL'17, pages 623–632. Association for Computational Linguistics.

El Hihi, S. and Bengio, Y. (1995). Hierarchical recurrent neural networks for long-term dependencies. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, NIPS'95, pages 493–499, Cambridge, MA, USA. MIT Press.

Haveliwala, T. H. (2002). Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA. ACM.

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

He, X., Kan, M.-Y., Xie, P., and Chen, X. (2014). Comment-based multi-view clustering of web 2.0 items. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 771–782, New York, NY, USA. ACM.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Karpathy, A., Johnson, J., and Li, F. (2015). Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078. International Conference on Learning Representaions.

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

Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA. MIT Press.

Liu, J., Wang, C., Gao, J., and Han, J. (2013). Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 252–260. SIAM.

Ren, Z., Liang, S., Li, P., Wang, S., and de Rijke, M. (2017). Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 485–494, New York, NY, USA. ACM.

Saveski, M. and Mantrach, A. (2014). Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 89–96, New York, NY, USA. ACM.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., and Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 83–92, New York, NY, USA. ACM.