Explainable Recommendation System Using Aspect Based Sentiment Analysis

Yuval Heffetz
Ben Gurion University
Be'er Sheva, Israel
yuvalhef@post.bgu.ac.il

Danielle Kapon
Ben Gurion University
Be'er Sheva, Israel
kapond@post.bgu.ac.il

ABSTRACT

In recent years, the use of textual reviews of products in commercial sites has significantly increased, making meaningful information available to recommender systems. With the advance in NLP tools, it is now possible to extract users' sentiments from reviews toward items' aspects. In this work we suggest utilizing this additional information by incorporating it in a matrix-factorization (MF) based model, to result in more explainable recommendations that usually cannot be achieved due to the latency of the factorization concepts. We base our suggested model on the Explicit Factors Model (EFM) that generates recommendations by matching user's interests with item's features, extracted from the reviews corpus. Our suggested model, the EFM++ uses a new optimization algorithm for the factorization process of the EFM, allowing the simultaneous MF of three dependent matrices. We offer additional alterations of the model such as adding the popularity of items as additional aspect, and also a variety of measures to increase the quality of explicit-features rating matrices, included in the MF model, to improve the learning process. Offline experimental results on a Yelp dataset, demonstrate the advantages of our framework and enhancements to the EFM over MF models on both rating prediction and top-K recommendation tasks

Yuval Heffetz and Danielle Kapon. 2018. Explainable Recommendation System Using Aspect Based Sentiment Analysis. . , 9 pages.

1 INTRODUCTION

Many recommendation system algorithms are based solely on numeric ratings users give to items. Some of these algorithms, such as Collaborative Filtering (CF) based algorithms use Latent Factors in order to infer more information from the rating data, using different mathematical techniques [13]. While these methods often achieve great accuracy, the factor latency makes the reasons for a recommendation being given to a user, unexplainable. Explaining a recommendation has many advantages, such as increasing a users' trust in the system, enhancing the persuasiveness of the recommendation, helping users make better decisions, etc.[10]

In recent years, the use of written reviews, in addition to the star rating, has increased. A written review offers a bundle of information that can be extracted and used in recommendation systems, and provide great help in explaining recommendations. The free-text reviews can also assist with assessing the aspects of an item that a user is most interested, and that items are better at, using sentiment towards the different aspects. Aspect based sentiment analysis (ABSA) [6] can help with generating more meaningful recommendations and for gaining a better understanding of the users and of the items. Moreover, and most importantly, ABSA can assist with understanding the explicit features of the recommendation, which could be highly beneficial both for improving recommendation algorithms and for the explainable recommendation task.

The main idea behind the model suggested in this paper is using the ABSA as a part of a matrix-factorization algorithm, trying to take advantage of the high accuracy of this method while adding explainability. Incorporating the sentiment of users towards specific items' aspects in MF models has been the target of a few works in recent years. Some have tried integrating a language model with a rating matrix factorization model and a topic model, inferring the sentiments towards aspects as a part of the combined learning process[5]. While presenting good results, the complexity of the model is an important issue, resolved in the model suggested here by separating the language model to the pre-process stage. Others have offered the use of context, extracted from the reviews text, offered as a part of a recommendation explanation (i.e. "you might want to try the pasta dish") bringing a superior user experience though still not able to exceed the achievements of other traditional algorithms [2]. The use a graph representation of the relationship between a user, an item and an aspect combined with a MF model is another line of works [7] achieving high accuracy however it lacks the ability to provide clear explainable recommendations.

One of the more prominent models developed in this line research is the Explicit Factor Model (EFM)[15]. Based on the assumption that users care about different aspects of the items and items excel in different aspects, it uses an aspect based sentiment lexicon extracted from the reviews to capture the user-feature attention and item-feature quality. It integrates these into a matrix factorization model in order to produce rating prediction, and uses a ranking method taking into account user-item similarity for the top-K recommendations, as well as for explainable recommendations. The uniqueness of this method is that is uses both latent and explicit features (based on the extracted aspects) in th MF process, and it does so with higher accuracy over existing models. This model will serve as the basis for this paper's suggested model.

One main challenge of the EFM is the learning process of the model's latent factors, used for three matrix factorizations simultaneously, i.e. factorizations of the user-item matrix, item-feature matrix and user-feature matrix. The purpose of the process is to align the latent concepts vectors with the explicit features matrices. In this paper we suggest a new optimization process, based on stochastic gradient descent for this tri-factorization problem, allowing handling missing values in the matrices in a better way than in

, ,

[15], and the adding of a bias factor. This improvement results in an improved model referred as EFM+.

The EFM's user-feature and item-feature matrices constitute the users and items profiles and are constructed by the attention users give to aspects and the sentiments towards them. Items' ranking is based on user-item profiles similarity. In this paper we introduce the popularity of items to the recommender in a new way - as an additional feature that is added to these matrices, without discrediting coverage. The purpose is to match users that tend to like popular items with this kind of items and match users with unique taste to high-rated and not so popular items. In order to further enhance the accuracy of the explicit-features-based similarity we introduce some more improvements such as advanced polarity analysis of the sentiments so it will capture a range of sentiment rather than just binary analysis (i.e. positive/negative), a different rescaling method for inducing a higher variance in the explicit features matrices' values and changing the similarity measure to cosine similarity.

This work will include a background chapter [section 2] which will include details about the explored research field, as well as an intro to the EFM model; a Related Work [section 3] chapter which will review the related work in depth; the Method [section 4] chapter which will include the details of our suggested improvements for the EFM model, to finally creating the EFM++ model; an Evaluation chapter [section 5] which will introduce an evaluation plan of the EFM++ model for each of the recommendation tasks; a Results chapter [section 6] which will present the results of the offline experiments that were conducted on the EFM++ model; a Discussion chapter [section 7] which will include a discussion over the results received from the offline experiments; and finally a conclusions chapter[section 8].

2 BACKGROUND

Our field of research is a sub-field of the more general Explainable Recommendation systems, that contains many different methods and models which include the use of Matrix Factorization models; Topic modeling; Graph Based models; Deep Learning; Knowledge-Base Embedding; Data Mining and Post Hoc. For this research, we will be focusing on explainable recommendation systems which integrates Matrix Factorization and Collaborative Filtering models, while focusing on models utilizing text reviews using ABSA [6]. The main publication which we will be focusing on (Zhang, Yongfeng, et al, 2014) addresses the problem of the explain-ability of recommendation systems by presenting a new model called the Explicit Factor Model (EFM)[15]. This will be our base for the development of our model, the EFM++.

The method proposed, suggests enabling explainable recommendation while sustaining and even outperforming the accuracy of state of the art factorization models. The main idea of the EFM is based on the assumption that different users are likely to care about different aspects of items. The model captures this orientation of a user and tries to match it with an item which performs better in the aspects that are in the user interest. This is done by integrating explicit factors in a factorization model. The EFM focuses on datasets where the ratings of items consists of an overall rating as well as a written free-text review. The explicit features are being extracted from the text reviews' corpus. A sentiment lexicon is built to capture the sentiment of each user towards each feature/aspect of the item. The lexicon is of the form (F, O, S), where F represents a feature word, O represents an opinion word and S represents the sentiment towards the feature, S = {negative: -1; Positive: +1}. After the lexicon is built, the user attention (interests) and the item qualities on the features are captured in two matrices. We will not focus on the construction of the lexicon in this paper, since it is not the main problem we are trying to solve.

The main advantage of the model is that in addition to the rating prediction and top-k tasks, the model generates an explanation of the recommendation in a fixed template. The explanation consists of the feature which was the most influential in the recommendation process. The model was evaluated both offline, on two different datasets, as well as online. It achieved superior results to other state-of-the-art models in the rating prediction task as well as in the top-k task. Moreover, the model increased the persuasiveness of recommendations in the online experiment, thanks to the explanations generated. Another advantage of the model is the ability to explain a dis-recommendation which is unique to this model.

3 RELATED WORK

Another interesting model (Diao, Qiming, et al., 2014) is the Jointly Modeling Aspects, Ratings and Sentiments (JMARS)[5].

The claim of the paper's authors is that in factorization models using only the numerical ratings, latent concepts parameters are invariant under rotation. If the prediction of a rating is done using $r_{um} \sim < v_u, v_p >$, the relative orientation of the vectors in space doesn't alter the outcome. This means that the meaning of the factors is hard to understand. The model is using the aspect based sentiments in the review corpus to solve this problem. Unlike the EFM, JMARS does not use a prepared sentiment lexicon but rather learns the sentiments towards different aspects in the reviews as a part of an integrated model. It uses the ratings to predict the aspect based sentiments as part the statistical model. It assumes that a review score arises from the process of combining partial scores associated with different aspects of an item.

The model is quite complex and it is comprised of a few different sub-models which are integrated together – the interests of the users and the aspects of the items are modeled together in a sub-model which is integrated with a ratings MF model and with a topic model. In addition, a language model which is comprised out of 5 smaller language models for each word group (Background words, sentiment words, movie-specific words, aspect-specific words and aspect specific sentiment words) is also integrated in the model, as a part of the topic model, which is used to generate words out of the language models.

Unlike EFM, the JMARS model does not use latent factors in addition to the explicit factors, which can result in a less flexible model and cause overlooking factors that may influence the ratings. Furthermore, the current JMARS model suffers from a long runtime due to the inference algorithm that is used. Moreover, we believe that integration of a few sub-models such as in JMARS may not always be a positive idea due to its high complexity, and that ABSA can be done separately as presented in the EFM.

The Sentiment Utility Logistic Model (SULM)[2], extracts features (i.e., aspects) and user sentiments regarding these features

(Bauman et al., 2017). The features and sentiments are integrated into a matrix factorization model to fit the unknown sentiments and ratings, which are finally used to generate recommendations. The proposed method not only provides recommended items to users, but also provides the recommended features for an item, and the features serve as the explanations for a recommendation. The key difference of this method from previous ones presented in this survey is the fact that the method can recommend restaurants, for example, together with more specific recommendations. These include the most important aspects over which the user has control and can potentially select them such as the time to go to a restaurant, e.g. lunch vs. dinner, and what to order there, e.g., seafood. It does so by understanding the user's personal choices regarding the different aspects in the past. Another difference is the focus on the evaluation of user experience. Although the model presents a new and important feature and increases the user experience, it does not achieve significantly better results in the basic task of rating prediction.

Another method that uses ABSA to give top-k explainable recommendations, is TriRank[7] (He, Xiangnan, et al., 2015). The model uses a tripartite graph, with vertex sets that represent users, items and aspects, and edges that represent the relationship between the vertices. The input includes triplets which indicate that a user mentioned a certain aspect in his review on a certain item, and this input creates an edges triangle in the graph. The weights on the edges denote the significance of the relationship between two entities (similar to EFM, the significance of a user-aspect relationship can be the number of times it was mentioned in the review, and the significance of a user-item relationship is the rating). The model uses genetic algorithm for ranking the vertices of tripartite graph by regularizing the smoothness and fitting constraints (while attempting to retain the prior belief). The model achieved better results of the Yelp and Amazon Electronics datasets (by Hit Rank and NDCG metrics), over 5 algorithms, even with noisy data. TriRank's incorporation of aspects also provides users with more transparency into the recommender system behavior and affords user interaction to further improve recommendations. For the goal of our research, TriRank is missing some functionality, such as rating prediction, as well as a clear explanation to the user regarding the reason for each recommendation.

4 METHOD

In this section the EFM++ method will be presented in comparison to the baseline model, the EFM. First, a quick overview of the baseline model's main framework components will be presented and the improvements of the EFM++ will follow.

4.1 EFM FRAMEWORK OVERVIEW

User-Item matrix A. Each entry in this matrix is the rating given by user $u \in \{1, 2, ..., m\}$ to item $i \in \{1, 2, ..., n\}$.

Aspect-based sentiment lexicon. $\mathcal{L}$. As presented in the background section, the lexicon is constructed by analyzing the entire review corpus for aspect-based sentiment analysis with NLP tools. Each entry of the lexicon is of the form (F,O,S) where F is a feature (aspect) word (e.g. food), O is an opinion word (e.g. good) and S is the sentiment towards the aspect which can be 1 for positive or -1 for negative.

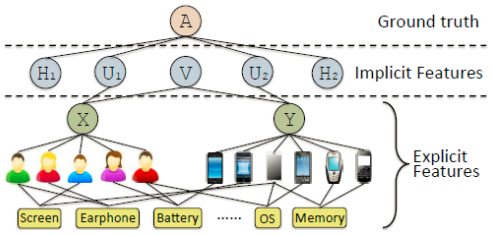

Figure 1: Relations between the ratings and the explicit features through the hidden parameters

Explicit features matrices X and Y. X is the User-Feature Attention Matrix where each entry in it expresses the interest of each user u in feature $f \in \{1, 2, ..., p\}$ by normalizing the number of times the user mentioned the feature in his reviews to the range of [1,N], N being the maximum rating in A. Y is the Item-Feature Quality Matrix expressing the average sentiment of the users towards each feature also normalized to [1,N].

Matrix Factorization of A, X and Y. The purpose of this stage is integrating matrices X and Y in a factorization model along with A. Unlike previous Matrix Factorization models, the learned vector parameters for predicting the missing values in A will be aligned in the vector-space according to the explicit features matrices at the end of the optimization process, allowing more accurate predictions and explainable recommendations. The learned parameters, i.e. the hidden factors, are modeled as follows - $U_1 \in \mathbf{R}^{m \times r}$ and $U_2 \in \mathbf{R}^{n \times r}$ are the hidden parameters matrices for predicting the missing values of A while capturing the user-attention matrix and item-feature matrix representations. We will refer them as the *Explicit Factors* matrices. r is the number of explicit factors. The EFM creators acknowledge that parameters representing the explicit features are not enough and so for the task of predicting A's missing values, they added the matrices $H_1 \in \mathbf{R}^{m \times r'}$ and $H_2 \in \mathbf{R}^{n \times r'}$ which are supposed to capture other hidden concepts. We will refer them as *Latent Factors*. r' is the number of latent factors. The relations between the different factors and parameters are presented in Figure 1.

After optimizing the different parameters, the estimates of A, X and Y can be made as follows:
Overall ratings - $A \cong U_1 U_2^T + H_1 H_2^T$
User-Feature attention - $X \cong U_1 V^T$
Item-Feature quality - $Y \cong U_2 V^T$

4.2 EFM+ (SGDTF OPTIMIZATION)

This section will survey the main contribution of our paper, stochastic gradient descent based optimization for tri-factorizations (SGDTF).

Let A, X and Y be the user-item ratings matrix, user-feature attention matrix and item-feature quality matrix respectively, constructed with accordance to the EFM framework presented briefly

in chapter 4.1 and in details in [15]. Let U_1, U_2, H_1, H_2 and V be the latent parameters of the integrated factorization model.

The optimization process for learning the latent parameters so that they will be aligned according to the explicit features while successfully predicting A has to be simultaneous. The learning process of the model should result in three different matrix factorizations that are dependent on the same latent factors, simultaneously, i.e. factorizations of the user-item matrix, item-feature matrix and user-feature matrix. The loss function should be computed in respect to the values of the three matrices in each iteration (see the overall relations between to model's components in fig.[1]). The integrated Matrix Factorization optimization task can be summarized with the following equation:

$$\underset{U_1,U_2,V,H_1,H_2}{\text{minimize}} \{\left\|PQ^T - A\right\|_F^2 + \lambda_x \left\|U_1V^T - X\right\|_F^2 + \lambda_y \left\|U_2V^T - Y\right\|_F^2 + \lambda_u \left(\|U_1\|_F^2 + \|U_2\|_F^2\right) + \lambda_h \left(\|H_1\|_F^2 + \|H_2\|_F^2\right) + \lambda_v \left(\|V\|_F^2\right)\}$$
$$\text{s.t.} U_1 \in \mathbf{R}_+^{m\times r}, U_2 \in \mathbf{R}_+^{n\times r}, V \in \mathbf{R}_+^{p\times r}, H_1 \in \mathbf{R}_+^{m\times r'}$$
$$H_2 \in \mathbf{R}_+^{m\times r'}, P = [U_1H_1], Q = [U_2H_2]$$

where the λ's are regularization coefficients.

The optimization algorithm used in the EFM that can be seen in [15] is based on a method for orthogonal nonnegative matrix tri-factorizations developed by C. Ding et al. [15]. The algorithm optimizes the objective function with respect to one parameter (i.e. hidden factors matrix) while fixing the other four parameter matrices in each line of the algorithm. The algorithm is based on complex mathematical computations that will not be detailed in this paper.

The main limitation of this algorithm is its lack of ability to deal with missing values. The user-item ratings matrix is very sparse and most of its entries are empty, expressing either dis-interaction between a user and an item or an interaction that the user chose not to rate. while the latter might be transformed to a negative rating, the former most possibly could not be interpreted in any way. Assuming some value should be put in the missing entries, the suggested optimization algorithm will learn the parameters in respect to the values put in these entries too, since the optimization is done for the whole matrix at a time, with no possibility to differentiate the missing values. The EFM framework does not provide a solution for this issue.

The solution for this problem suggested in this paper is optimization based on stochastic gradient descent (SGD), as presented in [15] with modifications for the tri-factorization problem. The optimization problem is in-fact divided into three smaller problems - (1) optimizing parameters with respect to A (2) optimizing parameters with respect to X (3) optimizing parameters with respect to Y. Although SGD is a method for advancing iteratively to a minimum point for one objective equation, it can also be used for the tri-factorization problem with the adding of a hyper-parameter, β, which will control the common conversion criteria for the three smaller problems presented above, as can be seen in our suggested optimization algorithm, alg.[2].

The convergence is defined as the point where both $RMSE_A$ and $MEAN\_RMSE$ are not descending simultaneously. The challenge is finding the best hyper-parameters that will allow the convergence to occur at the common minimum point for both $RMSE_A$ and $MEAN\_RMSE$. The β parameter purpose is to control the weight of each of the three factorization problems in the mean-RMSE computation. Lower values will give more weight to X and Y resulting in greater accuracy in the similarity part of the ranking formula that will be described in chapter 4.3 and larger values will result in greater rating prediction accuracy. The challenge is to find a value good both for ranking and for rating prediction. The γ parameters control the learning rate.

Algorithm 2: SGD optimization for tri-factorization

input : $A, X, Y, m, n, p, r, r', \beta, \gamma_a, \gamma_x, \gamma_y, \lambda_a, \lambda_x, \lambda_y$
output: U_1, U_2, H_1, H_2, V
repeat
  for *each user u and item i where* A_{ui} *non-empty* **do**
    $e_{ui} \leftarrow A_{ui} - (U_{1u} \cdot U_{2i}^T + H_{1u} \cdot H_{2i}^T + \mu + b_u + b_i)$
    $b_u \leftarrow b_u + \gamma_a \cdot (e_{ui} - \lambda_a \cdot b_u)$
    $b_i \leftarrow b_i + \gamma_a \cdot (e_{ui} - \lambda_a \cdot b_i)$
    $U_{1u} \leftarrow U_{1u} + \gamma_a \cdot (e_{ui} \cdot U_{2i} - \lambda_a \cdot U_{1u})$
    $U_{2i} \leftarrow U_{2i} + \gamma_a \cdot (e_{ui} \cdot U_{1u} - \lambda_a \cdot U_{2i})$
    $H_{1u} \leftarrow H_{1u} + \gamma_a \cdot (e_{ui} \cdot H_{2i} - \lambda_a \cdot H_{1u})$
    $H_{2i} \leftarrow H_{2i} + \gamma_a \cdot (e_{ui} \cdot H_{1u} - \lambda_a \cdot H_{2i})$
  end
  for *each user u and feature f* **do**
    $e_{uf} \leftarrow X_{uf} - (U_{1u} \cdot V_f^T)$
    $U_{1u} \leftarrow U_{1u} + \gamma_x \cdot (e_{uf} \cdot V_f - \lambda_x \cdot U_{1u})$
    $V_f \leftarrow V_f + \gamma_x \cdot (e_{uf} \cdot U_{1u} - \lambda_x \cdot V_f)$
  end
  for *each item i and feature f* **do**
    $e_{if} \leftarrow Y_{if} - (U_{2i} \cdot V_f^T)$
    $U_{2i} \leftarrow U_{2i} + \gamma_y \cdot (e_{if} \cdot V_f - \lambda_y \cdot U_{2i})$
    $V_f \leftarrow V_f + \gamma_y \cdot (e_{if} \cdot U_{2f} - \lambda_y \cdot V_f)$
  end
  $RMSE_A \leftarrow rmse(A_{validationSet}, U_1, U_2, H_1, H_2, V)$
  $RMSE_X \leftarrow rmse(X_{validationSet}, U_1, V)$
  $RMSE_Y \leftarrow rmse(Y_{validationSet}, U_2, V)$
  $MEAN\_RMSE \leftarrow \beta \cdot RMSE_A + \frac{1-\beta}{2} \cdot (RMSE_X + RMSE_Y)$
until *MEAN_RMSE and* $RMSE_A$ *both converge*;

As can be seen in Algorithm [2], changing the optimization algorithm to SGDTF from the algorithm suggested by [15] enabled the adding of the bias factor, b_u for each user u and b_i for each item i, defined in respect to the mean rating μ, as explained by [10].

Since the EFM with the optimization algorithm suggested by [15] did not result in adequate enough results, the EFM with Algorithm [2] will be considered as the new baseline model for the rest of the paper and will be referred to as EFM+.

4.3 EFM++

The model suggested in this paper, EFM++, is based on the EFM+ framework structure but incorporates improvements on top of it. In this chapter, a comprehensive survey of the EFM++ framework stages, as described in fig. [2], will be presented.

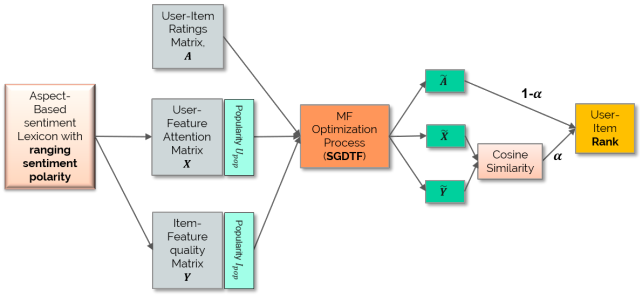

Figure 2: EFM++ Basic Framework Stages

Aspect-Based Sentiment Lexicon

The first step in the process of preparing the dataset for the model is the construction of the aspect-based sentiment analysis (ABSA) lexicon $\mathcal{L}$. It is constructed with the TEXT-API [1] NLP tool. Each review in the review corpus is analyzed and for each aspect mentioned in the review an entry is added to the lexicon. The analysis of each review results in a list of mentioned aspects (features) and the polarity (positive, negative) towards each aspect. In addition, to each aspect the confidence (probability) it actually appears in the text is assigned and also the confidence the suggested polarity is correct. The polarity confidence can be seen as the 'strength' of the polarity towards the aspect. Therefore, an entry is in the form of $(feature, feature\_confidence, polarity, polarity\_confidence)$. Each entry is associated with the user u that mentioned the feature and the item i the feature was mentioned about.

Explicit Features Matrices X and Y

As explained in chapter 4.1 the purpose of the user-feature attention matrix X is to capture the extent each user cares about each feature. It does so by counting the number of times each user mentioned each feature, t_{uf}, according to the ABSA lexicon. This number is rescaled to the range of [1, N] where N is the maximum rating in A. The item-feature quality matrix Y is capturing the sentiment towards each feature in respect to each item. It rescales to the same range [1, N], the value $k \cdot s_{if}$ where k is the number of times the feature f was mentioned in the reviews of item i and s_{if} is the average sentiment towards this feature in these k mentions.

Rescaling X and Y. The EFM method for rescaling t_{uf} and $k \cdot s_{if}$ values is by a reformulation of the sigmoid function as described in [15]. The problem with this kind of rescaling is the lack of variance in the transformed data. If the user mentioned a feature more than twice, the transformed value will always be between 4.0 and 5.0, meaning that a feature that was mentioned for example 20 times will have almost the same 'rating' in matrix X as a feature that was mentioned only 3 times by the user.
The EFM++ uses a different and straight-forward rescaling method that resulted in greater variance of the transformed values, the linear transformation. The resulting rescaling formulas are as follows.

$$X_{uf} = \begin{cases} 0, \text{ if user } u \text{ did not mention feature } f \\ \frac{t_{uf} - \min_{k \in F}\{t_{uk}\}}{\max_{k \in F}\{t_{uk}\} - \min_{k \in F}\{t_{uk}\}} \cdot (N-1) + 1, \text{ else} \end{cases}$$

In this formula, F is defined as the set of features containing $f \in \{1, 2, ..., p\}$. For the Y matrix we had to perform another modification other than just rescaling $k \cdot s_{if}$. A simple linear transformation would have resulted in a situation where at least one of the features would get the maximum rating N for each item even if the average sentiment towards that feature is not as high. The solution is to rescale in respect to the highest average sentiment to any feature out of all the items, defined by the set I.

$$Y_{if} = \begin{cases} 0, \text{ if item } i \text{ is not reviewed on feature } f \\ \frac{s_{if} - \min_{j \in I, k \in F}\{s_{jk}\}}{\max_{j \in I, k \in F}\{s_{jk}\} - \min_{j \in I, k \in F}\{s_{jk}\}} \cdot (N-1) + 1, \text{ else} \end{cases}$$

Popularity as Explicit Feature. Another limitation of the EFM is that the explicit-features matrices are constructed solely on the features extracted from the reviews corpus. It is possible in some datasets to obtain additional explicit features from outside sources which in turn can be incorporated in matrices X and Y, constituting users and items profiles in a manner that resembles content-based recommendation systems, where a user profile is matched with an item profile.
In some cases there is no additional available data on the items. The EFM++ offers a possible additional feature which is always available - the popularity of each item. Treating the popularity as an item feature is a unique idea, but one that can offer great improvement in the ranking task.
Popularity-based recommendation systems are very hard to beat but they offer no personalization and very low coverage of the items set, which are both very important features of a recommendation system. The EFM++ is not suggesting giving greater weight to popular items, but rather wishes to match users that like popular items with these kind of items and users that tend to like more unique and less popular items with the unpopular but high rated items.
Adding this feature is done by adding a column vector, $Ipop^{n \times 1}$ to the item-feature quality matrix Y which will hold the number of reviews given to each item (see fig.[2]), rescaled to [1, N] with respect to the highest number of reviews ever given to an item:

$$Ipop_j = \frac{review\_count_j - \min_{i \in I}\{review\_count_i\}}{\max_{i \in I}\{review\_count_i\} - \min_{i \in I}\{review\_count_i\}} \cdot (N-1) + 1$$

Now, the popularity feature could be computed for each user as the average popularity of the relevant items the user reviewed. Relevant items can be defined as items that were rated above a specific threshold by the user. The set of indexes of this items for each user will be defined by Rel_u. The additional column vector to matrix X is $Upop^{m \times 1}$:

$$Upop_u = \operatorname*{average}_{i \in Rel\_u}\{Ipop_i\}$$

Sentiment Polarity. The EFM uses a sentiment lexicon where the polarity is measured with a binary value of 1.0 for a *positive* sentiment and -1.0 for a *negative* sentiment. This method does not capture the full range of possible sentiment that can be felt by a user towards an item's aspect.
Using the lexicon constructed with the TEXT-API tool, this range of sentiment can be extracted with a probability computation using the

$feature\_confidence$ and the $polarity\_confidence$. The probability a feature was mentioned in a review text along with the strength of the sentiment polarity felt towards that feature can result in a range of sentiment between -1.0 and 1.0:

$$s = polarity \cdot feature\_confidence \cdot polarity\_confidence$$

where s is a sentiment towards a feature, $feature\_confidence \in [0, 1]$, $polarity\_confidence \in [0, 1]$ and $polarity \in \{-1 : negative; 1 : positive\}$. We anticipate that this action will result in a greater variance of the values in the item-feature quality matrix which will improve the learning process of the latent factors and further enhance the accuracy of the similarity measure in the rank computation.

Optimization and Prediction. The model's parameters optimization process is similar to that of the EFM+ and was detailed extensively in chapter 4.2. The first task is the prediction of A, X and Y values according to the description in chapter 4.1.

Ranking for Top-K Recommendations. Top-k recommendations list for each user is constructed with a score given for each item by a ranking formula. Similarly to the EFM method, The EFM++ score for each item is based on the assumption that a user decision for purchasing an item is taken by considering a few features of the item he cares about the most. For user u, let the column indexes of the k largest values in row vector $\tilde{X}_i$ ($\tilde{X}$ being the predicted user-feature interest matrix), be $C_u \in \{c_{u1}, c_{u2}, ..., c_{uk}\}$. Since the EFM++ has an additional feature, the popularity feature, we shall enforce the index of this feature in C_u by $C_u \cup p$, (p being the index of the popularity-feature column). The ranking formula is a trade-off between a similarity score for a user-item pair based on the explicit-features matrices and the direct user-item predicted rating, as can be seen in the suggested equation:

$$R_{ui} = \alpha * \left(\frac{\sum_{c \in C_u} \tilde{X}_{uc} * \tilde{Y}_{ic}}{\sqrt{\sum_{c \in C_u} \tilde{X}_{uc}^2} \cdot \sqrt{\sum_{c \in C_u} \tilde{Y}_{ic}^2}} \cdot (N-1) + 1 \right) + (1-\alpha)\, \tilde{A}_{ui}$$

Using the EFM ranking formula [15] while enforcing the popularity feature could end up in favoring the more popular items and thus causing lower coverage, which is not a desirable result. The similarity part of the EFM formula did not normalize the dot-product of the user and item vectors with the actual values of the vectors. A better similarity measure, used by the EFM++ is the cosine similarity. Using the cosine measure, linearly rescaled to the range of the user-item rating matrix [1, N] as presented in the previous equation is both a solution for the discussed problem and results in a better ranking, as will be seen in the results chapter.

Explainable Recommendations. The final task is to generate a personalized feature-level explanation. The ability to list the features that play the biggest role in the recommendation is the advantages of the EFM++. The model also provides a side bonus in the form of explaining a dis-recommendation. In a recommendation, the feature F_c is presented to user u_i as the main reason for the recommendation where:

$$c = \text{argmax}_{c \in C_i} \tilde{Y_{jc}}$$

In other words, the selected feature is the best performing feature of the item out of the most cared-of features of the user. Since the EFM++ could not be tested online for the affect of the explanations, no improvements were added over the EFM. However, we would suggest listing the k most cared features of the user with the predicted rating for each feature. This could be done quite easily and we believe it could increase the explain-ability.

5 EVALUATION

In this section, we will review the evaluation plan for EFM++, that is conducted by an off-line experiment. The data was collected from Yelp dataset, and contains written reviews and ratings for restaurants in Toronto, Canada. The dataset contains 3,063 users, 586 items and 49,928 reviews. The dataset contains at least 5 reviews per user and 10 reviews per restaurant. The range of ratings that can be given to the restaurants in the data set is [1, 5] with an increment of 1. The reviews corpus included mentions of 14 different aspects - food, location, ambiance, service, drinks, etc.

5.1 Experimental Plan

In order to try and follow the spirit of the evaluation plan of the EFM paper to the best of our abilities, an offline experiment was conducted. The tasks in review are the rating prediction task, and the top-k recommendation task. The hypotheses are (1) the prediction accuracy has improved over the baseline model and (2) the top-k recommendations list quality is also improved.

Data Splitting. For the offline experiment, the data was split into a train set and a test set on a temporal basis. For each user, the list of his ratings (reviews) was sorted according to their attached time-stamp. The first 80% of the list were considered as a part of the train set and the remaining 20% were moved to the test set. As can be seen in alg. [2] a validation set is also being used as a part of the optimization process, and it is extracted out of the train set in the same manner the test set was constructed, i.e. by time-stamp. We believe that this splitting method will best simulate an online scenario and thus it is better for the purpose of our experiment.

Evaluation Metrics. To evaluate the results of the rating predicion task, we used the Root-mean-square (RMSE) measure:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \hat{y_i} - y_i}{N}}$$

Where y_i is the true rating, $\hat{y_i}$ is the predicted rating, and N is the number of predicted items. This measure will be used to compare the rating prediction ability of the EFM++ with the baseline model, EFM+. We will also compare the results to the performance of other traditional FM models as well, while keeping as much common hyper-parameters, such as number of latent factors, identical. Since the EFM++ is a MF model, we shall choose for the comparison other MF models, namely SVD++ [10], NMF [8], and SlopeOne [4] that are known to excel in the ratings prediction task. All models were trained over the same train set and tested over the same test set.

Regarding the top-K recommendations task, the offline experiment will include the recommendation of the k highest ranked items to each user. The list is generated by ranking only the items in the test set that were actually given a rating by the user and not in respect to all possible items [15]. The quality of the recommendation list will be evaluated with two measures. The results will be compared to the baseline model and to the same MF models detailed

before, in addition to the BPRMF which is a 'learning to rank' MF model. The first measure is Normalized Discounted Cumulative Gain (NDCG@K):

$$NDCG_p = \frac{\sum_{i=1}^{p} \frac{2^{rel_i}-1}{log_2(i+1)}}{\sum_{i=1}^{|REL|} \frac{2^{rel_i}-1}{log_2(i+1)}}$$

Where rel_i is the true rating assigned by the user to the item located in the $i'th$ spot of the recommendations list. *REL* is the set of relevant items in the dataset (ordered by true rating) up to index p, which is the equivalent to k in the top-K task. Relevant items are defined as items that were rated 4.0 and 5.0 by the user. This metric penalizes the model if irrelevant items are recommended to the user in a higher rank than relevant ones.
The second measure is *precicion@k* which is computed as the ratio between relevant items and the total number of items in the recommended list.

Hyper-parameters. Choosing the right hyper-parameters is crucial for the success of the EFM++ model, especially with the suggested optimization algorithm. Since the data was split by time and the train and test sets are fixed, the parameters were not chosen with k-fold cross-validation. However the learned parameters are initialized randomly each run, so different results may be received. For this reason, the hyper-parameters were chosen by running the experiment 3 times with each set of hyper-parameters. The set of hyper-parameters that resulted in the best average results was chosen: $k, \alpha, r, r', \beta, \gamma_a, \gamma_x, \gamma_y, \lambda_a, \lambda_x, \lambda_y$.
The values used for each hyper-parameter will be presented in the next chapter. The presented results are an average of 10 reputations of each experiment.

6 RESULTS

The results of the offline experiment, as set out by the experimental plan, will be surveyed in this chapter.

6.1 Rating Prediction

The purpose of this evaluation is to measure the accuracy of the rating predictions by the EFM++. The importance of the rating prediction task is debated in recent years, when some argue predicting ratings accurately does not necessarily testify a good recommendation list. Nonetheless, this measure is still customary, especially for the evaluation of MF models.

Optimization Process. Before evaluating the prediction accuracy with RMSE, we would first like to validate the optimization process suggested in this paper for EFM+, the SGD optimization for tri-factorization. As explained in chapter 4.2, the learning process of the model's parameters should be optimized in respect to the values in three different matrices, X, Y and A, measured by $RMSE_X, RMSE_Y$ and $RMSE_A$. In each iteration these are computed over the validation set. Looking at fig. [3], it is clear that with the right hyper-parameters bringing the three optimization components to convergence in the same iteration is achievable. In each iteration, some of the parameters (i.e. U_1, U_2) are updated so that they will be able to predict both the values of the user-item ratings matrix and the values of the explicit-features matrices. We can see in fig. [3] that at some points the optimization is struggling to find the correct path

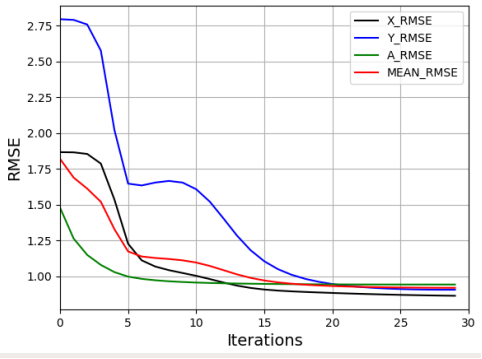

Figure 3: RMSE of the three different optimization components in the SGDTF process

Table 1: RMSE comparison over test set

NMF	SlopeOne	SVD++	EFM+	EFM++
1.088	1.057	0.996	0.995	**0.993**

for all three components but eventually aligns the rating prediction parameters with the explicit features, allowing accurate predictions for all three components.

Rating prediction. The set of hyper-parameters that achieved the best results for the EFM++ is as follows:
$r = 40, r' = 25, \beta = 0.5, \gamma_a = 0.005, \gamma_x = 0.0025, \gamma_y = 0.005, \lambda_a = 0.02, \lambda_x = 0.001, \lambda_y = 0.001$.
In accordance with the results of the EFM presented in [15], the ratio between the number of explicit factors, r' and the total number of factors, $r' + r$, that gave the best results is in the range of 30%-80%, which reinforce their conclusion that too many explicit factors may ruin the predictions.
As detailed in chapter 4.2, the purpose of β is to control the weight of each of the optimization components in the mean RMSE computation that determines the convergence criteria. The β value that was used allowed the continuation of the optimization process even when $RMSE_Y$ went up giving more weight to $RMSE_A$ in the *MEAN_RMSE* computation, as can be seen in fig. [3]. It was tolerant enough to the ascending while still giving enough weight to the explicit features matrices components.
In table [1], which presents the results of the experiment using the RMSE metric over the test set, we can see that both EFM+ and EFM++ performed better than the FM models NMF, SVD++ and SlopeOne. The EFM++ is superior over the EFM+, using the best possible hyper-parameters in each model.

6.2 Top-k Recommendations

In this section the quality of the recommendations list given to each user is evaluated by the NDCG@K and *precicion@K* measures, as detailed in the experimental plan. The evaluation was performed on a top-5 recommendations list.

NDCG Results. The results as presented in table [3] prove the superiority of both EFM+ and EFM++ over other MF models. In this

Table 2: Precision@5 comparison over test set

NMF	SVD++	SlpoeOne	BPRMF	EFM+	EFM++
0.732	0.760	0.747	0.635	0.769	**0.777**

Table 3: NDCG@5 comparison over test set

NMF	SVD++	SlpoeOne	BPRMF	EFM+	EFM++
0.713	0.736	0.727	0.651	0.750	**0.754**

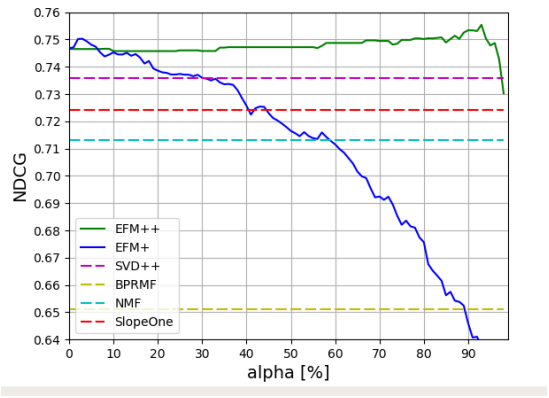

Figure 4: P@5 comparison with changing alpha

task too, the EFM++ achieved the best results, with a statistically significant improvement over the strongest competitor, the SVD++ model. The best results of the EFM++ were achieved with $k = 2$ (in the rank equation from chapter 4.3) meaning that the user-item similarity was most accurate when computed in respect to the two most important aspects for the user. Similar results were achieved in the precision measure, as can be seen in table [2].

Importance of user-item similarity . Hyper-parameter α determines the trade-off between the user-item similarity and the predicted rating in the ranking equation, as explained in chapter 4.3. Larger α means more weight to the user-item similarity part of the equation. In fig. [4] we can see that while in the EFM+ the user-item similarity weight is very small when the NDCG is at its peak, the EFM++ present a much more stable behavior, and a value of $\alpha = 0.93$ for the best possible NDCG. This means that the improvements incorporated in the EFM++, that targeted the explicit-features matrices and the similarity measure proved to be very beneficial. The quality of the recommendation list and its explainability has improved with the EFM++ since the ranking is relying more on the common aspects the user care about and the item excels at.

7 DISCUSSION

Both the EFM+ model and EFM++ model proved to be successful and improve the results for both the rating prediction task and the top-K recommendation task. The EFM++ was proved to be favorable to both the SVD++ and EFM+ baselines.

The new SGDTF optimization algorithm has proved to be applicable for the tri-factorization problem while allowing biases into the EFM+ model, greatly improving the results.
The improvements of the explicit features part of the model, namely the rescaling of the X and Y matrices to a linear scale rather than an exponential scale, the replacement of the similarity measure in the ranking equation to the Cosine Similarity, the adding of the popularity feature to the profiles of the users and the items and the ranging of the sentiment polarity, all together constituting the EFM++, made a significant impact on the quality of the recommendations list and on its reliance on the explicit-features similarity score, making it much more stable.

Moreover, while not detailed here due to lack of space, it is important noting that each of these modifications improved the results on its own. Relating to the popularity as an aspect seems to have a good effect on the model results while not harming coverage, meaning that users consider the popularity aspect of a restaurant when thinking of dining there. Increasing the variance of the explicit-features matrices by the rescaling method allowed for more efficient learning process and Adding polarity range captured better the users sentiment.

8 CONCLUSIONS

In this paper, we proposed the utilization of the ever-growing reviews corpus in recommendation system datasets by analyzing the sentiments of users towards different item aspects with text-analysis tools. Relying on the Explicit Features Model (EFM), we extract explicit product features and user opinions from reviews, then incorporate both user-feature and item-feature relations as well as user-item ratings into a unified hybrid matrix factorization framework aimed at the factorization of three rating matrices, all depended on the same latent factors. For this tri-factorization task we proposed a new SGD based learning algorithm allowing improved rating predictions, especially by dealing better with missing values and by incorporating user and item bias factors to form the new EFM+.

We then add another aspect to each user and item profile, the popularity aspect, and incorporate it into the user-item similarity score, as well as introducing different rescaling measures and the use of sentiment polarity range for greater variance in the explicit features data and finally we use the cosine similarity measure as a part of the ranking method for the top-k recommendation task, to form the improved EFM++ model.

The offline experiments show that our framework is superior to the EFM+ baseline and to different MF models in two tasks: rating prediction, and top-K recommendation. In addition, the different improvements resulted in a more stable hybrid model that can rely on the explicit-features' user-item similarity in the task of item ranking and list recommendation and for generating explanations and understand the reasons behind the MF process.

This is a another step towards integrating ABSA for explainable factorization models, and there is much room for improvements. Future work can focus on (1) incorporating additional features from outside sources in addition to the aspects extracted from reviews and (2) explanations presentation such as a list of the per-aspect rating for each item.

REFERENCES

[1] Aylien TEXT ANALYSIS API. [n. d.].

[2] Bauman, Konstantin, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM* (2017).

[3] C. Ding, T. Li, W. Peng, and H. Park. 2006. Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering. *KDD* (2006), 1236–135.

[4] D. Lemire and A. Maclachlan. 2005. Slope one predictors for online rating-based collaborative filtering. *Proceedings of the SIAM Data Mining Conference (SDM '05)* (2005).

[5] Diao, Qiming, et al. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM* (2014), 478–510.

[6] Feldman, Ronen. 2013. Techniques and applications for sentiment analysis. *Commun. ACM* (2013), 82–89.

[7] He, Xiangnan, et al. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM* (2015).

[8] Lee, D. D., and Seung, H. S. 2001. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* (2001).

[9] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. 2011. Lexicon-Based Methods for Sentiment Analysis. *Computational Linguastics* (2011).

[10] N. Tintarev and J. Mastho. 2011. Designing and Evaluating Explanations for Recommender Systems. *Recommender Systems Handbook* (2011), 478–510.

[11] Ricci, F., Rokach, L., and Shapira, B. . 2015. Recommender systems: introduction and challenges. (2015).

[12] S. Rendle, C. Freudenthaler, et al. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *UAI* (2009).

[13] Su, Xiaoyuan, and Taghi M. Khoshgoftaar. 2009. Algorithms for non-negative matrix factorization. *Advances in artificial intelligence 2009* (2009).

[14] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: An optimization approach. *WWW* (2011).

[15] Zhang, Yongfeng, et al. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval. ACM* (2014).