Information Systems 83 (2019) 1–12

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is

Combining different metadata views for better recommendation accuracy

Rafael M. D'Addio *, Ronnie S. Marinho, Marcelo G. Manzato

Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil

HIGHLIGHTS

- Two items representations based on BabelNet concepts extracted from users' reviews.
- Combination of two item views of features: one statistical and one based on quality.
- Three combination categories: pre, neighborhood and post combination.
- Post combination provides the best results with approach based on machine learning.
- Neighborhood combination has promising results especially with larger neighborhoods.

ARTICLE INFO

Article history:
Received 24 January 2018
Accepted 18 January 2019
Available online 6 February 2019
Recommended by Laks Lakshmanan

Keywords:
Recommender systems
Item representation
Unstructured metadata

ABSTRACT

Recommender systems emerged as means to help users deal with information overload by filtering content based on their preferences. Regardless of the recommendation method, there has been a recent interest in using user reviews as source of information, since they contain both detailed items' descriptions as well as users' opinions. Even though several works have been done in the subject, very few of them consider different views that the items may have towards their features, selecting only a method of weighting their features. In this work, we propose a system that combines two item representations that represent different views of the same feature set: one based on its statistics and the other based on its quality. Features are disambiguated concepts extracted from users' reviews. We propose several strategies divided into three categories: pre combination, neighborhood combination and post combination. We evaluate our strategies in two data sets, comparing them with each other and against the isolated item representations, as well as a representation baseline based on terms and sentiment analysis. Results are promising showing that some combinations are capable of producing better rankings than their isolated versions.

1. Introduction

Recommender systems nowadays are largely used as tools that aid users in their decision-making processes regarding what to use or consume, by filtering content according to their preferences. They can be classified into two important paradigms: content-based filtering and collaborative filtering, which determine how to evaluate data and make suggestions [1]. Content-based filtering recommends items to a user based on information relevant to the items. In this approach, items are represented based on their available content: metadata, synopses, reviews, among others. Users, in turn, have a profile that is inferred from the items they have evaluated, and the recommendation occurs by combining the profiles with the item representations. In collaborative filtering, in turn, only user evaluations and interactions are used to define relationships between themselves and/or between items. Hybrid methods, in turn, have gained prominence by combining aspects of both paradigms, minimizing recurrent problems in both [1].

Regardless of the method, there has been a growing effort to exploit user reviews for better descriptions of items or users in recommender systems [2]. Reviews are a great source of information since they contain relevant descriptions about items, as well as personal opinions regarding the item and its features.

Works that use reviews direct their efforts to better describe items [3,4], users [5,6] or both [7], but they propose simple feature extraction approaches that do not consider text ambiguity problems such as synonymy and polysemy. Given that, other works focus on solving those issues, but apply their efforts on impersonal texts such as synopses and Wikipedia[1] descriptions [8–14],

* Corresponding author.
E-mail addresses: rdaddio@icmc.usp.br (R.M. D'Addio), ronnieshida@usp.br (R.S. Marinho), mmanzato@icmc.usp.br (M.G. Manzato).

https://doi.org/10.1016/j.is.2019.01.008
0306-4379/

[1] https://www.wikipedia.org/.

discarding information about users' opinions which is important in recommender systems.

Furthermore, such works do not consider the different views that each item may have towards their features. In other words, they use a single method to give weights for their characteristics, based on some statistics, such as the frequency of the term, or whether certain feature is present or not in the item; or based on any quality quantification, such as sentiment analysis. A system would benefit greatly if it could describe its items in more than one way such that each representation's strength could be explored towards producing better recommendations.

In response to these limitations, we propose in this paper a strategy that combines two different item representations in a recommender system. The representations, based on user reviews, are constructed from features directly extracted from text by employing a technique that extracts disambiguated concepts. They are constructed with the same set of concepts, varying in the weighting process: (i) a statistical view, with a TF-IDF-based weighting [15] and (ii) a quality-based view, using sentiment analysis [16] to assess whether the items' features are good (positive) or bad (negative). While the first view captures similarities within the content of items, the second computes items' similarities based on opinions of other users. For instance, the statistical view may regard items as similar if they have the same specific and/or rare features mentioned with similar frequency on their reviews, while the quality-based view may regard them as similar if their features have closer levels of appreciation/depreciation by the reviewers (e.g. two movies can be similar if their suspense is praised while the lightning is not).

Finally, in order to explore the strengths of both representations, we propose and analyze several combination strategies, which are divided into three categories: pre combination, neighborhood combination and post combination. We compare them against each other and against the isolated representations, as well as a baseline representation based on simple terms weighted by sentiment analysis [3].

The remainder of the paper is structured as follows. Section 2 gives a comprehensive overview towards related works; Section 3 presents preliminary concepts required to understand our proposal; Section 4 details the construction of the item representations used in this study; Section 5 details the combination strategies; Section 6 details the experiments conducted in this study; and, finally, Section 7 presents the conclusions and future work.

2. Related works

In this section, we present some related works. We begin presenting some approaches that deal with the ranking problem, then describe works that use information from unstructured data to aid in recommendation.

Correctly ranking suggestions is an ongoing problem in recommender systems. Perhaps the most classic approach in learn-to-rank is the Bayesian Personalized Ranking (BPR) strategy, which was proposed by Rendle et al. [17]. It uses pairs of items, where one is known to the user and the other is not, as the basis for a Bayesian optimization criterion. More details on this method can be seen in Section 3.2. On the same note, Yagci et al. [18] proposed two shared memory lock-free parallelized pairwise learn-to-rank approaches based on BPR. In this work, among the item representation combinations, we propose two BPR-based learn-to-rank approaches as a post-processing step, in order to calculate weights to combine the rankings produced by the representations applied in isolation to the recommender system.

Beyond learn-to-rank, some works try to aggregate additional information into their models. For instance, Liang et al. [19] proposed CoFactor, a co-factorization model that simultaneously factorizes the user–item implicit feedback and item–item co-occurrence matrices, producing item embeddings that further aid the recommendation. The co-occurrence matrix is obtained from the implicit feedback itself, thus not categorizing as an external information.

Several works aggregate information from unstructured data (reviews, synopsis, comments, among others) into recommender systems. Chen et al. [2] conducted a survey related to recommender systems that make use of user reviews. In the article, authors divide the works into two main categories: those that use reviews to generate item representations, and those that use them to generate and/or increase user profiles.

Recent works that use reviews to describe items tend to extract features to characterize them according to their characteristics. For example, in previous work [3], different features extraction techniques were used to define feature and aspect-based item representations, which were applied in a neighborhood-based algorithm. The weights of the characteristics for each item were calculated considering the average sentiment observed in their reviews. In a different approach, McAuley et al. [4] used user reviews to establish substitution and/or complementation relationships between items. To do this, they propose a model that extracts latent topics and uses them to predict the two types of relationships in directed graphs.

On the other hand, works that make use of reviews for the construction of user profiles can construct profiles of finer granularity which explain the preferences of the users in relation to the diverse characteristics of the items. An example is the work of Ganu et al. [5] which uses opinion mining in a restaurant recommendation system based on soft clustering. Terzi et al. [6], in turn, proposed a user-based neighborhood algorithm, whose similarities are calculated based on user reviews. The authors explore six alternatives for similarity metrics, all based on the WordNet[2] taxonomy. Regardless of the choice of the metric, the similarity between two reviews is calculated from the average similarity among all possible pairs of terms. However, the authors do not use lexical disambiguation techniques, but instead adopt the best combination of all the possible meanings of the terms.

Some papers aim to describe both users and items. For example, in the work of Chen et al. [7], a matrix factorization model is proposed that makes use of reviews to rank the features of products that are most relevant to the users preferences. The recommendation is based on this ranking, which is combined with the quality of the characteristics in the items, measured through opinion mining, and latent factors based on ratings.

None of the aforementioned approaches deal with text ambiguity problems such as polysemy and synonymy. On the other hand, there are efforts to minimize these problems on recommender systems, but they focus majorly on impersonal texts such as synopses and Wikipedia descriptions.

The WordNet taxonomy and its word-sense disambiguation (WSD) algorithms have been actively explored in the last decade by content-based and hybrid recommender systems. For example, De Gemmis, Lops, and colleagues extensively explored a strategy of constructing item representations and user profiles based on WSD of item synopses, and applied it in a variety of contexts, such as user-based hybrid recommender systems [10], content-based systems intended for multilingual recommendation [11], among others. Such a strategy divides texts into compartments, e.g. title, authors, body text, etc., which are processed by a WordNet-based WSD algorithm to produce different representations, whose weights are the synsets frequencies. Capelle et al. [9] proposed a content-based news recommender system that makes use of WSD based on the WordNet taxonomy for the construction of

[2] https://wordnet.princeton.edu/.

item representations and user profiles. They also extend their SF-IDF metric, introduced in [8], incorporating in the calculation a similarity based on page count returned by the Bing search engine[3] when querying for named entities present in the text. The SF-IDF weighing is an extension of the traditional TF-IDF [15], with the difference that features are synsets rather than terms. Systems that use the WordNet senses repository are limited by the low coverage of named entities. Although WordNet contains some instances of people, places and companies, the taxonomy is not very broad.

On the other hand, some works exploit entity linking (EL) techniques to perform feature extraction. For example, Musto et al. [12] propose a content-based system that uses EL to produce item descriptions to support the system to generate context-sensitive recommendations. Several different entity linking techniques are used to extract concepts directly from the Wikipedia, which are expanded by extracting their immediate ancestors in the Wikipedia category tree.

Finally, efforts have been made to incorporate techniques that accomplish both WSD and EL in content-based recommendation systems, such as the work of Narducci et al. [13] and Oramas et al. [14]. In [13], the authors evaluated four different types of representations in the multilingual recommendation task, two based on Wikipedia and two based on BabelNet[4] [20]. In [14], the BabelFy[5] [21] tool is used to extract concepts related to music and sounds, which are used to automatically produce a knowledge base in the form of a graph. The item representations are constructed by mapping the graph in a vector space from two strategies: entity-based and path-based.

All the aforementioned works deal with only one type of item representation, built using a single feature extraction technique and weighting scheme. In a previous study [22], we found evidence that using more than one item representation may be beneficial for recommender systems. In that work, we extracted features from four different techniques (two based on term extraction, one based on named entity recognition and one based on topic hierarchy) and constructed binary item representations (an item has or not a feature), and tested them into two recommender algorithms. Combination of the results were performed by a ranking ensemble based on BPR.

Our work differs from the aforementioned since we analyze a vast amount of approaches for correctly combining two item representations that characterize items in two different views of the same set of characteristics: one statistical and one regarding the item's quality. The characteristics used in this study also provide more semantics to the representations, since they are disambiguated concepts instead of simple terms.

3. Preliminary concepts

In this section we present some preliminary concepts for understanding our approach. First, we present the recommender algorithm used in our experiments (Section 3.1). Then, we briefly describe the Bayesian Personalized Ranking [17] optimization strategy (Section 3.2), which is extended in two of our ranking combination strategies.

3.1. Item k -NN

The recommendation algorithm in which we employ our items representations is the well-known Item k-NN, a collaborative filtering approach which calculates a neighborhood for each item and is used to derive a score for an unknown user–item pair [23].

This algorithm uses a similarity measure to obtain the neighborhood, whose size is set through a parameter k. In traditional Item k-NN, the similarity is calculated through items representations based on user rating and/or interaction patterns. In previous works, we have replaced the rating-based representations with feature-based ones derived from text [3].

Several similarity measures can be applied, such as cosine and Pearson correlation coefficient. In our work, we decided to use the latter based on results of preliminary experiments. The Pearson correlation coefficient can be calculated as [23]:

$$p_{ij} = \frac{\sum_{n=1}^{k}(w_n^i - \overline{w_i})(w_n^j - \overline{w_j})}{\sqrt{\sum_{n=1}^{k}(w_n^i - \overline{w_i})^2}\sqrt{\sum_{n=1}^{k}(w_n^j - \overline{w_j})^2}}, \tag{1}$$

where $\overline{w_i}$ and $\overline{w_j}$ are the average values of the features of items i and j.

In the item recommendation scenario, the score $\hat{r}_{ui}$ of an item i to a user u is simply the sum of the similarities of all other items j in the neighborhood $N_u(i)$, which is an intersection between the k most similar items and the set of items that u has interacted with [17]:

$$\hat{r}_{ui} = \sum_{j \in N_u(i)} p_{ij} \tag{2}$$

This score does not represent a rating that a user may provide to an item; instead, it is a score that represents the likelihood that a user may enjoy such item. The collection of scores of unknown items are used then to construct a ranking of suggestions that is provided to the user.

3.2. BPR Learning

The Bayesian Personalized Ranking strategy proposed by Rendle et al. [17] addresses the problem of ranking optimization by considering in its calculation the relationship between a known item i and an unknown item j. In order to do that, a set D_k is constructed containing all possible triples (u, i, j), where i is the item known to user u and j is the unknown. With this set at hand, a generic BPR optimization criterion is formulated:

$$\text{BPR-Opt} := \sum_{(u,i,j) \in D_K} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \ , \tag{3}$$

where $\hat{x}_{uij}$ is a real value that captures the relationship between a triple $(u, i, j) \in D_k$ in a given model (e.g. matrix factorization) with Θ as parameters, and λ_Θ the regularization values for the model parameters. The $\hat{x}_{uij}$ value can be decomposed in:

$$\hat{x}_{uij} = \hat{r}_{ui} - \hat{r}_{uj} \tag{4}$$

The goal of the BPR Learning algorithm is to maximize BPR-Opt through stochastic gradient descent. In this sense, the authors suggest to use a bootstrap sampling approach with replacement to select triples on which the following update formula is performed:

$$\Theta \leftarrow \Theta + \eta \left(\frac{1}{1 + e^{\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right), \tag{5}$$

where η is a learning rate constant.

4. Item representations

In this paper we focus our efforts into combining two item representations created from user reviews. They differ mainly in the way that the features are viewed, since both representations use the same feature set in their construction but are weighted differently. In this sense, one representation focuses on the statistics of an item's features, while the other focuses on their quality.

[3] https://www.bing.com/.
[4] http://babelnet.org/.
[5] http://babelfy.org/.

We use two external natural language processing (NLP) tools to extract features from user reviews: (i) Stanford CoreNLP[6] [24] and (ii) BabelFy [21]. In the following subsections we detail the steps taken in constructing the representations.

4.1. Text pre-processing and feature extraction

First, we process the items' reviews with the well-known Stanford CoreNLP [24], a natural language processing toolkit that contains several NLP routines. There, we perform routines such as tokenization, lemmatization, stemming, parsing, part-of-speech (POS) tagging and sentence splitting. We also execute a sentiment analysis algorithm in this toolkit, whose results are used in one of our item representations (as it can be seen in Section 4.3). The output of this process are XML files structuring the texts into sentences and their components, as well as their parser and sentiment.

From there, we reconstruct the documents into plain text so they can be processed by the other NLP tool, BabelFy [21]. BabelFy combines the tasks of word sense disambiguation and entity linking, extracting from raw text babel synsets, which are disambiguated concepts from the BabelNet [20] knowledge database. BabelNet condenses into one big database entries from Wikipedia and WordNet, and provides links between them. The reason for reconstructing the documents is because we preserve the Stanford CoreNLP sentence splitting, which is necessary to match BabelFy's output with the sentiments produced by the previous tool, thus allowing us to build one of our item representations.

The documents processed by BabelFy are text files where words are replaced by disambiguated babel synsets. From there, we extract candidate synsets that will compose our feature set, or vocabulary. First, we maintain in a list only the synsets that are nouns. Since this set is still very large, we use a filter based on the item frequency (IF) [25], which is an adaptation from the traditional document frequency (DF) [15] where a set of item's reviews is viewed as a single document. Let F be the synsets (features) set and I the items' set, the item frequency IF_f of a feature f is:

$$IF_f = \sum_{i}^{|I|} k_{if}, \tag{6}$$

where k_{if} is equal to 1 if an item i contains the synset in at least on of its reviews, and 0 otherwise.

Having calculated the IF_f of each synset, the filter consists of removing those whose value is inferior to a given threshold. In a previous study [26], we determined that a good threshold is the value of 10. We adopt this value in this study. The remaining synsets after this step will constitute the vocabulary, which will be used to produce the items representations that are detailed in the next subsections. Both representations are modeled in a vector space, where each feature corresponds to a dimension in that space.

4.2. SF-IDF

The first item representation strategy used in this work aims to characterize items according to a statistical view of their features. As in our previous work [26], we use the strategy proposed by Capelle et al. [8], where features are weighted by the SF-IDF, an extension of the well known TF-IDF metric [15].

This technique analyzes the synset frequency in a single item (SF) and its rarity among the whole item collection (IDF). Formally, we denote as $|I|$ the number of items of the system, and a synset s may appear in $|i : s \in I|$ items. We also denote as $n_{s,i}$ the frequency of a synset s in an item i. The synset frequency $SF(s, i)$ can be defined as [8]:

$$SF(s, i) = \frac{n_{s,i}}{\sum_k n_{k,i}}, \tag{7}$$

where $\sum_k n_{k,i}$ corresponds to the total frequency of the k other synsets present in i.

The inverse document frequency, IDF_i, is:

$$IDF(s) = \log \frac{|I|}{|i : t \in i|}. \tag{8}$$

Finally, the SF-IDF of a synset–item pair can be given as:

$$SF - IDF(s, i) = SF(s, i) \times IDF(s). \tag{9}$$

4.3. Sentiment

The second item representation strategy focuses on measuring the quality of the features of an item. In this sense, each synset is weighted according to the overall sentiment (e.g. positive, negative or neutral) that users assign in their reviews toward them. Thus, an item is represented by the average sentiment of many users' reviews towards each of its characteristics.

In order to do that, we use the sentiment analysis algorithm available at the Stanford CoreNLP toolkit [16], obtaining the sentiment for each sentence of each review. The main reason for using a sentence-level sentiment analysis is that since all of the features extracted from the reviews are nouns, which in most cases have neutral sentiment, a natural approach to obtain their sentiment is to match with sentiments obtained from the context they were used in the text. Moreover, since features are extracted from a different tool than CoreNLP, and they are concepts instead of words, it is justifiable to use the sentiment of the sentences they occur.

The sentiment analysis algorithm [16] uses recursive neural networks models to build representations that capture the structure of the sentences, obtaining their sentiment based on the meaning of each of words. It classifies sentences in five sentiment levels: "Very Negative", "Negative", "Neutral", "Positive" and "Very Positive". We convert this classification into a $[1, 5]$ rating system, being 1 equals to "Very Negative" and 5 equals to "Very Positive".

We use the following strategy to obtain the sentiments of the synsets, adapted from previous works [3]. First, the system analyzes the feature set, then, for each item, reads the BabelFy output texts, finding and storing the sentences IDs that are related to each of the synsets. Then, it matches the IDs with those of the CoreNLP files, storing their corresponding sentiments.

After obtaining the sentences related to each feature, the next step is the sentiment attribution. For each feature of each item it is calculated the average sentiment of the related sentences. Thus, this value represents the collective level of appreciation or depreciation of a certain attribute of an item. This approach does not consider the level of confidence that a final sentiment score may have: a score calculated from several mentions to a feature from several different reviews is more reliable than a score calculated from a single mention. In order to address this, we devised a heuristic where we dampen the sentiment value, i.e., approximate it to 3 (the neutral value), if the number of times it was mentioned is lower than the global average number of mentions. Formally, a sentiment $S(s, i)$ of a synset s in relation to an item i can be dampen by the following formula, if its mentions are lower than the global average:

$$S(i, j) = \begin{cases} S(s, i) + log_{10}(n_s/N) & \text{if S(s,i)} > 3 \\ S(s, i) - log_{10}(n_s/N) & \text{if S(s,i)} < 3 \end{cases}, \tag{10}$$

[6] http://nlp.stanford.edu/software/corenlp.shtml.

where n_s is the number of times the synset was mentioned in the item's reviews, and N is the global average number of mentions. With this heuristic, we are able to dampen the sentiment strength to a maximum of 1, still preserving its original orientation.

Finally, a zero value indicates that an item simply does not have that feature.

5. Recommendation strategy: Combining representations

As mentioned earlier, this work explores strategies to accurately combine the two item representations described in the previous section. Those strategies were elaborated to be applied in one of the steps taken during the recommendation, which is performed by the item-*k*NN algorithm.

We explore several strategies which are grouped into three main categories: (i) pre combination, which combines the item representations before they are processed by the algorithm; (ii) neighborhood combination, which combines the *k*-NN produced by the two representations; and (iii) post combination, which combines the rankings produced by each item-*k*NN execution with each representation. Fig. 1 illustrates our proposal.

In the following subsections we detail every strategy implemented in this study.

5.1. Pre combination

The pre combination comprises two heuristics to combine directly the items representations, i.e., combine the two item's vectors. This is done before the recommendation process begins, and the new item representation feed the recommender algorithm, which runs normally. The two heuristics are:

- *Multiplication:* In this strategy, the scores of the two representations are multiplied. In this sense, for each feature f in the vocabulary, its final score w_{fi} for a determined item i is:

$$w_{fi} = w_{fi}^{S} * w_{fi}^{SF}, \tag{11}$$

  where w_{fi}^{S} and w_{fi}^{SF} are the scores provided by the sentiment and the SF-IDF representations, respectively. Since both representations have the same structure, i.e., they only differ in the weight that each feature has, there is no possibility that scores are nullified by zero multiplications. Thus, the resulting item representation cannot be more or less sparse than the originals.
- *Concatenation:* In this strategy, we produce a final representation by concatenating both representations. The concatenation is performed at each feature, i.e., each feature is designed to comprise two dimensions of the feature space. In this sense, the final representation will have twice the number of features of the original, thus aggravating its sparsity.

5.2. Neighborhood combination

The neighborhood combination comprises four strategies to combine the neighborhoods produced by each of the representations. In this sense, the item-*k*NN is executed separately for each item representation up to the point where it generates the neighborhood for an unknown user–item pair. Then, one of the strategies is employed to generate a final neighborhood, which will then be used to produce the pair's recommending score. The strategies are based on either one of the two set of operations:

- *Intersection:* the strategies based on this operation aim to maintain only the items which are present on both neighborhoods. Thus, given the sentiment neighborhood $N_u(i)^S$ and the SF-IDF neighborhood $N_u(i)^{SF}$, the final set is:

$$N_u(i) = N_u(i)^S \cap N_u(i)^{SF}. \tag{12}$$

  The two strategies based on this operation differ only in the score calculation of the neighbors. One of them maintains the highest score among the two neighborhoods, while the other calculates their average. A sorting algorithm can also be applied as a final step. No further cuts to respect the k size of the neighborhood is needed, since at the worst case (both sets are the same), they still have their sizes limited previously by k.
- *Union:* the two remaining strategies are based on this operation, aiming to provide a combined ranking of neighbors. Thus, the final set of neighbors can be viewed as:

$$N_u(i) = N_u(i)^S \cup N_u(i)^{SF}. \tag{13}$$

  In order to produce the union of the two neighborhoods, a constraint must be fulfilled: the resulting set must not be larger than the k limit. To guarantee that, we follow some steps. First, both neighborhood are sorted from highest to lowest similarity. Next, we combine both sets into a single one. Similarly as in the intersection approach, the two strategies based on union differ only in the calculation of the similarity of neighbors that are present in both sets: one maintains the highest value, while the other calculates an average score. Since we employ those strategies in situations where there may be items appearing in both sets, the output of the combination is not necessarily sorted. Thus, we sort it, producing a final ranking of neighbors. Finally, we make a cut in the kth position of the ranking, maintaining only the most similar neighbors for both representations.

5.3. Post combination

In this category, we explore strategies that aim to combine user rankings generated by item-*k*NN for each item representation. In this sense, first we run each representation separately in the recommender and, as a post-processing step, we combine the resulting rankings.

We explore four strategies that can be divided into two very different approaches. Two of them are based on heuristics, while the other two are based on machine learning. In the next subsections they are described in details.

5.3.1. Heuristic strategies

The heuristic strategies, similarly to the neighborhood combination ones, base themselves on set operations. Specifically, those strategies use the union operation. Given a ranking $R_S(u)$ for the sentiment representation execution of the item-*k*NN, and a ranking $R_{SF}(u)$ for the SF-IDF version, the final ranking $R(u)$ for a user u is:

$$R(u) = R_S(u) \cup R_{SF}(u). \tag{14}$$

In order to perform the union of two rankings, we follow the same steps performed in the neighborhood strategy. As rankings are already sorted, we start by combining both rankings. As before, we treat item repetitions in two different manners: (i) we consider only the highest value of ranking score; (ii) we sum the two values, since we follow the premise that if an item is present in both rankings, it probably will be more relevant to the user. With that, the output from the merge-sort algorithm may not be sorted, requiring us to perform it after its completion. Finally, we cut the ranking in a predetermined size (e.g. 10), since a large ranking may be overwhelming for the user.

We do not use intersection operations since there is a probability that the resulting ranking is very small, or even empty. In a worst case scenario, each ranking will contain a different set of items, resulting in an empty final ranking.

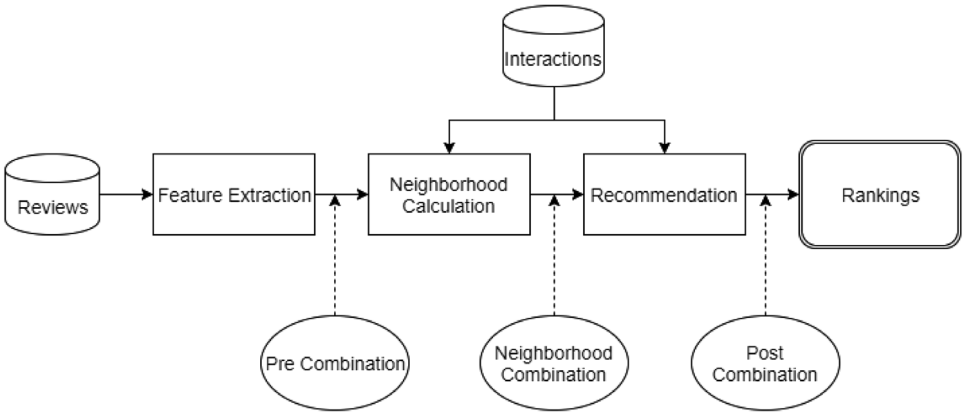

Fig. 1. The proposed system's architecture.

5.3.2. BPR Learning strategies

The machine learning strategies are based on BPR Learning, a concept we introduced in Section 3.2. Here, we use BPR to learn weights for a correct combination of two rankings. The idea, originally proposed in [27], modified the $\hat{x}_{uij}$ decomposition Eq. (4) into:

$$\hat{x}_{uij} = \alpha_a(\hat{r}^a_{ui} - \hat{r}^a_{uj}) + \alpha_b(\hat{r}^b_{ui} - \hat{r}^b_{uj}), \tag{15}$$

where α_a and α_b are weights assigned to predictions from recommender a and recommender b. In that work, in fact, the main goal was to use this strategy to combine multiple user interactions (such as ratings, purchase history and tag assignment), thus the equation above could be incremented with any amounts of α weights and their related predictions. The α weights were trained simultaneously with the recommender system, which was based on matrix factorization, resulting in the algorithm with full potential when using several different training data.

In our work, in turn, we restrict the calculation to only two sets of predictions. With this, a and b are not recommenders or types of interactions, but in fact two executions of the same recommender with different item representations. Moreover, we do not train any parameter of the recommender system, delegating the BPR training to a solely post-processing step. Thus, $\hat{r}^S_{ul}$ and $\hat{r}^{SF}_{ul}$ represent the predictions generated by sentiment and SF-IDF item representations (where l can be either i or j), as well as α_S and α_{SF}.

Applying the stochastic gradient descent, the update Eq. (5) can be employed, with the $\frac{\partial}{\partial\Theta}\hat{x}_{uij}$ component substituted by its corresponding derivative:

$$\frac{\partial}{\partial\Theta}\hat{x}_{uij} = \begin{cases} \hat{r}^S_{ui} - \hat{r}^S_{uj}, & \text{if } \Theta = \alpha_S \\ \hat{r}^{SF}_{ui} - \hat{r}^{SF}_{uj}, & \text{if } \Theta = \alpha_{SF} \end{cases}. \tag{16}$$

Another approach we experimented is to regard the known pairs $\hat{r}_{ui}$ as 1, since they are indeed always relevant for the user. With that, the Eqs. (15) and (16) can be rewritten as:

$$\hat{x}_{uij} = \alpha_S(1 - \hat{r}^S_{uj}) + \alpha_{SF}(1 - \hat{r}^{SF}_{uj}), \tag{17}$$

$$\frac{\partial}{\partial\Theta}\hat{x}_{uij} = \begin{cases} 1 - \hat{r}^S_{uj}, & \text{if } \Theta = \alpha_S \\ 1 - \hat{r}^{SF}_{uj}, & \text{if } \Theta = \alpha_{SF} \end{cases}. \tag{18}$$

Regardless of the approach, the process of learning the alphas is performed similarly as the original BPR Learning algorithm: we perform bootstrap sampling to select random triples from the vast amount of possible candidates and employ them to learn the *alpha* weights. After learning them, we multiply each ranking scores by their respective weights and then merge them both. In case of repetitions, i.e., both rankings have the same item, we sum their scores. It is important to note that the weights are not applied to all items, since the rankings are significantly smaller than the total number of items. Nevertheless, we experimented with several ranking sizes and determined that rankings of 100 items are optimal, since items beyond that position may not be very interesting for the final user. Finally, we sort the resulting ranking and perform a cut in a predetermined size (e.g. 10).

6. Evaluation

In this section we detail the experiments conducted in this study. In the following subsections we present the data sets used, the experimental setup and the results we obtained.

6.1. Data sets

We used two largely different data sets to evaluate our proposal.

The first is the well-known MovieLens 100k (ML-100k)[7] data set, which consists of 100,000 ratings (from 1 to 5) performed by 943 users for 1,682 movies. Each user of the data set has rated at least 20 movies. In order to perform our experiments, we collected up to 10 reviews per item from the IMDb[8] website, resulting in a total of 15,863 documents. The reviews were selected as the website's top-10, ordered by their helpfulness.

The second is one of the Amazon[9] data sets extracted by McAuley et al. [28].[10] They extracted many large data sets based on review data from Amazon, each representing a single domain of items: movies, books, electronics, among others. Given that most of them are extremely large, we selected the Apps for Android data set, which is the fifth biggest data set from the collection. The original data set has 2, 638, 172 ratings and 752,937 reviews from 1, 323, 884 users for 61,275 items, where users and items have at least 5 interactions. Given that it is a fairly large and sparse data set, we filtered it by maintaining only the reviews interactions and eliminated the items and users that had less than 10 interactions, resulting in 16,201 users and 4869 items, totaling 264,047 interactions.

As one can see, the two data sets are very different. To begin with, the user–item proportion is different, with the first data set having 1.78 times more items than users, while the second has 3.33 times more users. Furthermore, the second data set is significantly more sparse than the first: while it has 2.64 times

7 https://grouplens.org/datasets/movielens/100k/.
8 http://www.imdb.com/.
9 https://www.amazon.com/.
10 http://jmcauley.ucsd.edu/data/amazon/.

Table 1
Vocabularies sizes for the item representations regarded in this study.

	ML-100k	Amazon Apps
Heuristic terms	3085	3089
BabelFy	6238	8978
Pre comb. multiplication	6238	8978
Pre comb. concatenation	12,476	17,956

more interactions than the previous, its user–item matrix is significantly larger: 17.18 times more rows (users) and 2.89 times more columns (items).

6.2. Experimental setup and evaluation metrics

In order to check the strength of our proposal, we compare our work with another baseline item representation proposed in a previous work [3]. We also regard as baselines the isolated item representations based on BabelFy described in Section 4.

As for the baseline representation from [3], called *heuristic terms*, it is based on lemmatized terms extracted from the same reviews used in this study. These terms go through heuristics that filter by the part-of-speech tag, maintaining only nouns, and eliminate the less frequent terms, using IF (Eq. (6)) with threshold 30. Weighting of the terms is based on average sentiment, using the Stanford CoreNLP algorithm. Table 1 presents the sizes of the vocabularies of the item representations in both data sets, as well as those produced in the pre combination.

The baseline representations, as well as the combinations, were all executed in the item k-NN algorithm (Section 3.1), with $k = \{20, 40, 60, 80, 100\}$ and Pearson correlation coefficient (Eq. (1)) as the similarity measure. All rankings produced were of maximum size of 100 per user.

Regarding the post combination techniques, the cuts performed in the final rankings were also of size 100. The techniques based on BPR learning used learning rate $\eta = 0.05$ and regularization constants λ_S and $\lambda_{SF} = 0.0025$, defined experimentally.

We also compare our findings against the classical BPR-MF learn-to-rank approach [17], and two state-of-the-art ranking approaches. The first one, called CoFactor [19], perform co-factorization on both user–item implicity feedback and item co-occurrence matrices. The second, proposed by Yagci et al. [18], presents two parallel pairwise learn-to-rank approaches, called PLTRN and PLTRB. We apply those baselines in our experimental setting, using their own provided source code. We save their generated rankings and evaluate them in our evaluation protocol.

All baselines require hyperparameter tuning. For BPR-MF, PLTRN and PLTRB, we ran their implementations provided by Yagci et al. [18], with a fixed seed for random samplings. As for the parameters, we ran a grid search for learning rate η and number of factors F, obtaining $\eta = 0.005$ and $F = 20$ for ML-100k, and $\eta = 0.01$ and $F = 20$ for Amazon Apps. Regularization parameters were defined as $\lambda_{\theta_u} = \lambda_{\theta_i} = 0.0025$ and $\lambda_{\theta_F} = 0.00025$, as reported by the author. As for CoFactor [19], the authors define their hyperparameters by modifying them according to a scale ℓ, while also grid searching for the negative sampling value k. We follow this approach and grid search for both values, obtaining $\ell = 0.1$ and $k = 1$ for ML-100k and $\ell = 1$ and $k = 10$ for Amazon Apps.

We evaluated the quality of the rankings with the mean average precision at N (MAP@N) measure. It evaluates a ranking of size N, giving a greater weight for occurrences of relevant items in early positions of the ranking. It is a measure that produces a value which corresponds to the average of j queries, where each query produces a ranking and a score that is the average of different n precision levels. Formally, let $\{i_1, \ldots, i_{m_j}\}$ be the set of relevant items for a query $q_j \in Q$, and R_{jk} be the set of results returned from the first item until the i_k item, then the MAP can be measured as [15]:

$$MAP = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\frac{\#(\text{relevant items in } R_{jk})}{R_{jk}}. \quad (19)$$

We evaluate our results in four different N values of MAP: 1, 5 and 10. We do this in order to evaluate if our solution is capable of returning relevant items as the ranking increases in size, up to 10.

All experiments were carried out in a 10-fold cross validation setting. The data set was randomly divided into 10 folds, where each fold contained at least one interaction of each user. We define as training set 9 out of the 10 folds, and use the remaining as test. We iterate 10 times this procedure, varying the fold used on the test set. We report as results the average values of the iterations. In order to check their significance, we applied the two-tailed Wilcoxon signed-ranks test [29].

6.3. Results and discussion

In this section we present and discuss the results obtained by our approaches. Table 2 presents all results obtained in the experiments. All approaches and corresponding descriptions are summarized as follows:

- Pre comb 1: The pre combination approach detailed in Section 5.1 with the concatenated representations.
- Pre comb 2: The pre combination approach detailed in Section 5.1 with the multiplied scores of the representations.
- Neighbor comb 1: The neighborhood combination approach detailed in Section 5.2 with the intersection between the neighbors, using average values of similarity.
- Neighbor comb 2: The neighborhood combination approach detailed in Section 5.2 with the intersection between the neighbors, using the highest value of similarity.
- Neighbor comb 3: The neighborhood combination approach detailed in Section 5.2 with the union between the neighbors, using average values of similarity in case of repetitions.
- Neighbor comb 4: The neighborhood combination approach detailed in Section 5.2 with the union between the neighbors, using the highest value of similarity in case of repetitions.
- Post comb 1: The post combination approach detailed in Section 5.3 with the union between the rankings, considering the highest ranking score in case of repetitions.
- Post comb 2: The post combination approach detailed in Section 5.3 with the union between the rankings, summing ranking scores in case of repetitions.
- Post comb BPR 1: The post combination approach detailed in Section 5.3 with the BPR Learning algorithm considering the known pairs' predictions as 1.
- Post comb BPR 2: The post combination approach detailed in Section 5.3 with the BPR Learning algorithm using the known pairs' predictions.

In Table 2, numbers in bold represent the best results for each column, i.e. the best result for determined metric in determined number of k. Values with the symbol $*$ represent statistical significance with $p \leq 0.05$, and with the symbol $**$ represent significance with $p \leq 0.01$, both over all other approaches and baseline representations. In the following subsections we detail the results of each data set separately, and, finally, provide a discussion summarizing them.

6.3.1. Movielens 100k

Here we detail the results related to the MovieLens 100k data set. We first compare the results between each category of combination approaches, and then compare the best of each of them against each other.

Table 2
Results obtained for both data sets. Values with the symbol $*$ represent statistical significance with $p \leq 0.05$, and with the symbol $**$ represent significance with $p \leq 0.01$ over all other approaches and baselines.

MovieLens 100k															
	k = 20			k = 40			k = 60			k = 80			k = 100		
	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10
Isolated Representations															
Heuristic Terms	0,15917	0,22473	0,22001	0,14613	0,21481	0,20968	0,15631	0,22197	0,21587	0,15854	0,22068	0,21387	0,15684	0,21839	0,21391
Babelfy SF-IDF	0,15037	0,22472	0,22366	0,14793	0,22092	0,22226	0,14666	0,22010	0,22046	0,14783	0,22052	0,22019	0,14952	0,22131	0,22035
Babelfy Sentiment	0,20117	0,26343	0,25457	0,19576	0,25911	0,25052	0,19427	0,26097	0,25233	0,18961	0,25742	0,24979	0,18335	0,25168	0,24390
Pre combinations															
Pre comb 1	0,20530	0,26765	0,25685	0,19067	0,25575	0,24777	0,19300	0,26027	0,25146	0,19162	0,25869	0,25036	0,18441	0,25196	0,24385
Pre comb 2	0,14995	0,22065	0,22100	0,14719	0,21769	0,21769	0,15143	0,21975	0,21877	0,15483	0,22230	0,22048	0,15589	0,22300	0,22043
Neighborhood combinations															
Neighbor comb 1	0,16253	0,18043	0,17910	0,14467	0,19467	0,19316	0,15899	0,21362	0,21114	0,16939	0,22140	0,21683	0,15878	0,21100	0,20773
Neighbor comb 2	0,16497	0,18345	0,18332	0,13460	0,18687	0,18550	0,15189	0,20913	0,20724	0,16939	0,21998	0,21574	0,15581	0,20866	0,20635
Neighbor comb 3	0,15663	0,23323	0,22969	0,17614	0,24891	0,24292	0,17847	0,25078	0,24577	0,17678	0,24743	0,24204	0,17200	0,24397	0,23925
Neighbor comb 4	0,17413	0,25175	0,24566	0,18844	0,26326	0,25555	0,19332	0,26642	0,25943	0,19046	0,26289	0,25622	0,19024	0,26072	0,25467
Post combinations															
Post comb 1	0,15546	0,23437	0,23086	0,17381	0,24430	0,23955	0,18505	0,25440	0,24838	0,18632	0,25438	0,24795	0,18282	0,25187	0,24470
Post comb 2	0,14539	0,21861	0,21710	0,14422	0,21264	0,21118	0,13563	0,20343	0,20254	0,12057	0,18699	0,18883	0,11453	0,17882	0,18093
Post comb BPR 1	**0,21644**	0,28924	0,27681	**0,20933***	**0,28196****	**0,27239****	**0,21029****	**0,28421****	**0,27437****	**0,21039****	**0,28187****	**0,27154****	**0,20265***	**0,27389****	**0,26603****
Post comb BPR 2	0,21304	**0,28960**	**0,27950**	0,19618	0,27273	0,26619	0,19135	0,27143	0,26415	0,18706	0,26137	0,25616	0,17847	0,25281	0,24785
Amazon Apps															
	k = 20			k = 40			k = 60			k = 80			k = 100		
	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10	MAP@1	MAP@5	MAP@10
Isolated Representations															
Heuristic Terms	0,01865	**0,03461****	**0,03868****	0,01964	0,03488	0,03936	0,01816	0,03416	0,03882	0,01852	0,03444	0,03919	0,01876	0,03445	0,03929
Babelfy SF-IDF	0,01736	0,02817	0,03085	0,01831	0,03043	0,03333	0,01978	0,03248	0,03562	0,02070	0,03387	0,03717	0,02112	0,03456	0,03792
Babelfy Sentiment	0,01366	0,01979	0,02114	0,01489	0,02357	0,02558	0,01626	0,02767	0,02969	0,01564	0,02820	0,03162	0,01604	0,02921	0,03259
Pre combinations															
Pre comb 1	0,01475	0,02485	0,02782	0,01479	0,02715	0,03061	0,01512	0,02815	0,03199	0,01499	0,02800	0,03197	0,01497	0,02817	0,03222
Pre comb 2	0,01731	0,02808	0,03079	0,01850	0,03052	0,03350	0,02003	0,03249	0,03571	0,02042	0,03332	0,03661	0,02079	0,03413	0,03749
Neighborhood combinations															
Neighbor comb 1	0,00770	0,00773	0,00773	0,00636	0,00641	0,00641	0,00611	0,00627	0,00627	0,00610	0,00622	0,00622	0,00591	0,00606	0,00607
Neighbor comb 2	0,00671	0,00677	0,00677	0,00583	0,00587	0,00587	0,00579	0,00591	0,00592	0,00561	0,00567	0,00568	0,00624	0,00638	0,00638
Neighbor comb 3	0,01797	0,03067	0,03416	0,01899	0,03308	0,03693	0,01992	0,03458	0,03867	0,02000	0,03504	0,03936	0,02004	0,03525	0,03964
Neighbor comb 4	0,01923	0,03221	0,03559	0,02006	0,03434	0,03816	0,02108	0,03600	0,04004	0,02160	0,03689	0,04111	0,02153	0,03716	0,04156
Post combinations															
Post comb 1	0,01787	0,03011	0,03340	0,01913	0,03277	0,03634	0,02007	0,03434	0,03827	0,01984	0,03473	0,03888	0,02035	0,03515	0,03932
Post comb 2	0,01308	0,02352	0,02668	0,01376	0,02524	0,02870	0,01462	0,02655	0,03019	0,01466	0,02697	0,03086	0,01460	0,02699	0,03079
Post comb BPR 1	0,01857	0,03254	0,03600	0,02030	0,03560	0,03926	0,02189	0,03764	0,04166	0,02310	0,03948	0,04356	0,02308	0,03987	0,04407
Post comb BPR 2	**0,01992***	0,03329	0,03683	**0,02120***	**0,03609**	**0,03988**	**0,02277****	**0,03838****	**0,04224***	**0,02407***	**0,04013***	**0,04424****	**0,02435****	**0,04073***	**0,04490***

Table 3
Comparison between the best combination approaches (with $k = 20$) and the baselines for the MovieLens 100k data set..

	MAP@1	MAP@5	MAP@10
Pre comb 1	*0.20530	*0.26764	0.25685
Neighbor comb 4	*0.17412	0.25175	0.24566
Post comb BPR 1	***0.21643**	***0.28924**	***0.27680**
BPR-MF	0.13648	0.25237	0.26250
PLTRN	0.13669	0.25113	0.26202
PLTRB	0.14390	0.25584	0.26405
CoFactor	0.05429	0.10841	0.12288

Table 4
Comparison between the best combination approaches (with $k = 100$) and the baselines for the Amazon Apps data set..

	MAP@1	MAP@5	MAP@10
Pre combination 2	0.02079	0.03413	0.03749
Neighbor combination 4	*0.02152	0.03716	0.04156
Post combination BPR 2	***0.02435**	***0.04072**	***0.04489**
BPR-MF	0.02027	0.03589	0.04057
PLTRN	0.02028	0.03578	0.04059
PLTRB	0.02057	0.03584	0.04058
CoFactor	0.01620	0.02793	0.02916

Regarding the pre combination approaches, the item representations concatenation (i.e. Pre comb 1) showed the category's best results, being statistically superior with $p \leq 0.01$ against the multiplication approach. It also provided statistically superior results against two of the baseline representations: Heuristic terms and BabelFy SF-IDF. In relation to BabelFy Sentiment, it is statistically superior with $p \leq 0.05$ only for $k = 20$.

Concerning the neighborhood approaches, the neighbor union with highest similarity in repetitions (i.e. Neighbor comb 4) provided the best results, being statistically superior with $p \leq 0.01$ against all other approaches of this category, and the Heuristic terms and Babelfy SF-IDF baseline representations. In relation to BabelFy Sentiment, it has worse results with $k = 20$, it is not statistically different with $k = 40$ and it is superior with $p \leq 0.05$ with the remaining k.

Regarding the post combination results, both BPR Learning-based approaches provide the best results (being both statistically superior with $p \leq 0.01$ against the others of this category), but the version where known pairs' predictions are considered as 1 (i.e. Post comb BPR 1) is the best between them. This approach has no statistical significance against the other BPR Learning approach for $k = 20$, but it is superior with $p \leq 0.01$ for the rest of the results. In relation to the baseline representations, it is statistically superior than all of them with $p \leq 0.01$ in all cases, except for MAP@1 with $k = \{20, 40, 100\}$, where it is superior with $p \leq 0.05$.

Comparing each category's best approach, the post combination BPR Learning approach has the best results for this data set, being statistically superior with $p \leq 0.01$ in the majority of the cases, and $p \leq 0.05$ in the remaining. A graphical comparison between them and the baseline representations can be seen in Fig. 2.

Finally, we compare our representations with some state of the art recommender baselines. We elect as the best neighborhood configuration $k = 20$, and compare its results against the baselines in Table 3. Numbers with bold typeset represent the best results for each MAP value. Also, those with * are statistically superior with $p - value < 0.01$. As it can be seen, the Post comb BPR 1 approach produced the best results and it is statistically superior against all baselines. Pre comb 1 produced statistically superior results against the baselines for MAP@1 and MAP@5, while Neighbor comb 4 is statistically superior only on MAP@1.

6.3.2. Amazon apps

We detail in this section the results from the Amazon Apps data set in the same way as the previous data set.

In relation to the pre combination category, the results of the multiplication approach (i.e. Pre comb 2) are the best, being statistically superior with $p \leq 0.01$ against the concatenation approach. Despite being the best result for this approach, it does not have better results than the baselines, being inferior to Heuristic terms, and having very close results with BabelFy SF-IDF.

Regarding the neighborhood strategies, again the best approach is the neighbor union with highest similarity in repetitions (i.e. Neighbor comb 4), having statistically superior results with $p \leq 0.01$ against other approaches from this category. This approach has statistically better results (with $p \leq 0.01$) than all baseline representations with $k \geq 60$, except for MAP@1. With $k = 40$, the results are close to those from Heuristic terms, while $k = 20$ provide inferior results.

Concerning the post combination results, again both BPR Learning-based approaches provide the best results, being both statistically superior with $p \leq 0.01$ against the others of this category. Between them, the version where we use known pairs' predictions (i.e. Post comb BPR 2) can be considered the best, being statistically superior than the other BPR Learning approach with $p \leq 0.01$ in half the cases, and $p \leq 0.05$ in the rest. In relation to the baseline representations, it is inferior than Heuristic terms for $k = 20$, in metrics MAP@5 and MAP@10, and it has no statistical difference for $k = 40$ at MAP@5 and MAP@10. For the remaining k, as well as the remaining baseline representations, it is statistically superior with $p \leq 0.01$.

The post combination BPR Learning approach has the best results between the best approaches of each category, being statistically superior with $p \leq 0.01$ except for $k = 20$, where it is superior with $p \leq 0.05$ against the neighborhood approach. A graphical comparison between them and the baseline representations can be seen in Fig. 3.

Finally, we again compare our representations with the recommender baselines. We elect as the best neighborhood configuration $k = 100$, and compare its results against the baselines in Table 4. Numbers with bold typeset represent the best results for each MAP value. Also, those with * are statistically superior with $p - value < 0.01$. The Post comb BPR 2 approach produced the best results and it is statistically superior against all baselines. Pre comb 1 has superior, but not statistically different results against the baselines for MAP@1, while Neighbor comb 4 is statistically superior only on MAP@1, while nonetheless presenting better results for the remaining MAP values.

6.3.3. Discussion

As it can be seen, both BPR Learning strategies provide the best results in both data sets, but in each data set one has performed better than the other. With that, further experimentation in other data sets is required in order to define which is better. Nevertheless, a great advantage that Post comb BPR 1 has in relation to the other is that it requires less computational resource and is faster, since it does not require to compute predictions of the training pairs, considering them as 1. Even so, both proved to be good approaches, especially since their strength resides in using a learn-to-rank approach to correctly weight the relevance of each ranking.

The heuristics ranking approaches, in turn, did not provide good results. We argue that using direct combinations may have dragged relevant items to lower positions of the ranking. Since those approaches do not have a learning mechanism to weight what is or is not relevant, the resulting ranking has a poorer composition than the isolated rankings. Even so, the Post comb 1 approach was able to provide better results against baseline representations in some

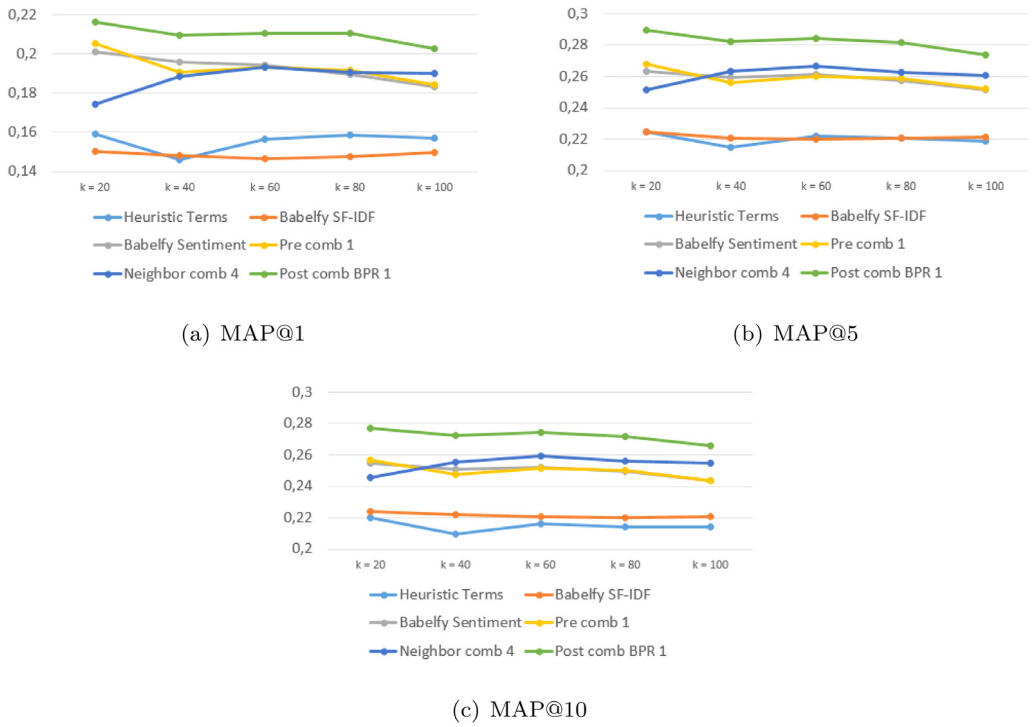

Fig. 2. Comparison between the best combination approaches and the baseline representations for the MovieLens 100k data set.

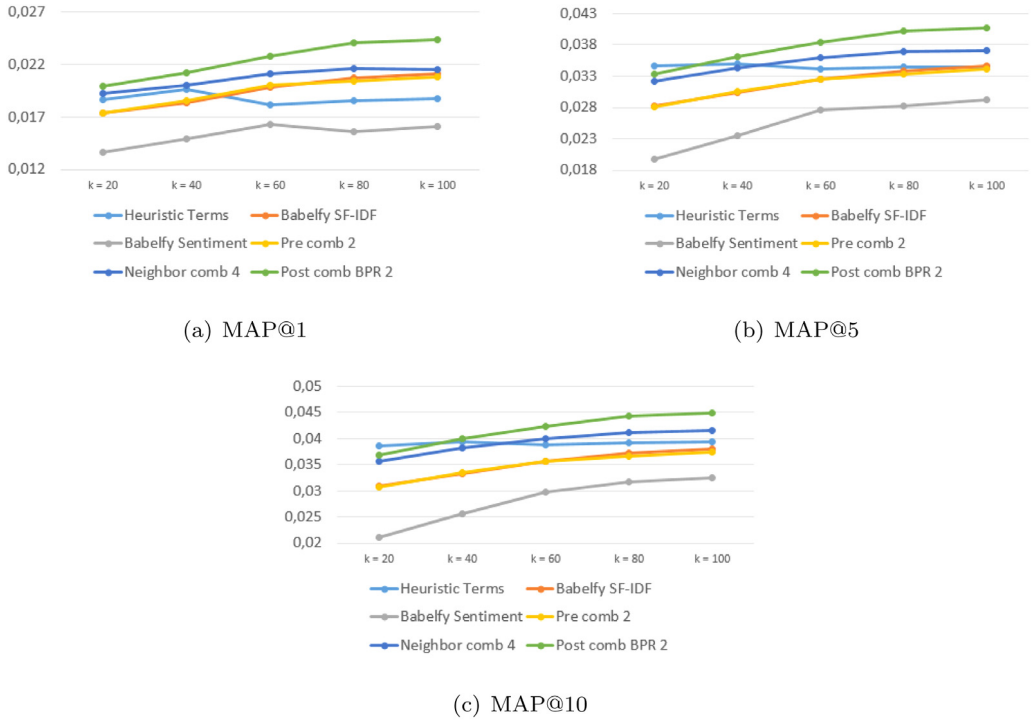

Fig. 3. Comparison between the best combination approaches and the baseline representations for the Amazon Apps data set.

cases, especially with larger rankings (MAP@5 and MAP@10) and larger neighborhoods ($k = \{80, 100\}$).

The pre combination strategies also did not have good results. As well as the BPR strategies, further experimenting is required to define which strategy is the best, since in each data set one performed better than the other. Furthermore, even though those strategies' results did not have statistical significance against the baseline representations, in several cases they performed slightly better than the isolated BabelFy representations that they are based on.

The neighborhood combination category results, in turn, were more consistent: in both data sets, the Neighbor comb 4 approach provided the best results. It also performed better than the baseline representations for $k \geq 60$. We argue that since we are combining neighbors, the larger the set, the better the combination can be performed. Since the sets are an intersection between the most

similar items and the set of items evaluated by the user, as we raise k we have more chances of constructing a neighborhood. This can be clearly seen with the intersection neighborhood combination approaches: they have very poor results since their final neighborhood sets are potentially very small. As for the strategy for combining repeated items' similarities, the best approach was to maintain only the highest value. We argue that the reason it performed better is that, in this sense, good neighbors are not dragged down by lower scores in the other neighborhood, thus maintaining their high similarity score, and, finally, not having a chance of being cut in the final sorting step.

Nevertheless, in most of the cases, each category produced better results than both the baseline representations and recommenders for at least one of their approaches. By comparing against the baseline recommenders, it can be noticed that in most cases both pre and neighborhood combinations produce better results, and at some cases with statistical difference, for MAP@1 and MAP@5. This highlights the ability of our approaches to produce better suggestions early on the rankings. Our BPR-based combination approaches also produce far superior results against baselines, highlighting the importance of combining two different views of the items.

This thorough evaluation shows the strength of our approaches, and allows us to select which of them to use according to our needs. Since the pre combination approaches require little computational resources and are only processed once before the recommendation, they can be used in any settings and still provide a small gain on ranking accuracy for the system. They can also be used on other content-based recommender systems, but further experiments are required to assess their advantages on these scenarios. The neighborhood combination approaches in turn revealed good results, specially with larger sets of k, being strongly recommended to be used in such scenarios. They also have the advantage of being simpler and faster alternatives than the BPR learning-based post combination approaches. These, in turn, have the best of the results, but are the ones that require more of computational resources, being suggested to be used on top notch servers. Moreover, as well as the pre combination approaches, these are not bound to the item-kNN algorithm, being capable of being used in any recommender system that can produce rankings.

Finally, the Amazon Apps results, in general, are very low. This can be attributed to the large size of the data set and the evaluation method itself. Since items can be considered relevant only if they are in the test set, and the total number of items is very large, it becomes difficult to correctly match items in such small ranking with those present in the test set.

7. Conclusions and future work

In this paper we have explored several strategies for correctly combining two item representations based on user reviews in a recommender system. Experiments showed promising results, with machine learning-based ranking combination strategies presenting the best results of all approaches. Below we summarize the main contributions of this paper:

- The item representations based on user reviews. These representations carry a large amount of semantics, since they use concepts instead of simple terms as features. The main implication is that concepts minimize problems of ambiguity that terms can have. Moreover, since the item representations are based on texts provided by users, they can be employed in any data domain, i.e., movies, music, products, among others.
- Two different views of the same feature set. This way, items can be detailed in two different manners in relation to their features: one that scores features according to their statistics in relation to the items, and the other that scores features according to the sentiment users have towards them in the items.
- The combination of these representations to improve recommendation accuracy. By combining the representations, the system is analyzing the items in two manners, and the strength of both can be exploited. We explore an extensive array of combination approaches and analyze their contributions to produce rankings.
- The generality of pre and post combination strategies. In this paper we explored these approaches in an item-kNN algorithm, but they are not bound to this algorithm. They can be employed with other content-based recommender, but further experimentation must be carried out to assess their advantages.

As future work, we intent to perform more experiments in data sets of other domains and sizes in order to define which techniques are the best among those that could not be defined in this study. We also aim to explore machine learning to give weights for pre and neighborhood combinations. Finally, the semantics of the BabelNet synsets extracted from the texts can also be explored to further enrich the representations, since the synsets are connected in a very large network with semantic relations that are derived from both WordNet and Wikipedia.

Acknowledgments

The authors would like to thank the financial support of: CAPES, Brazil, CNPq, Brazil and FAPESP, Brazil (process 2016/20280-6).

Conflict of interest

Declarations of interest: none

References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. GutiéRrez, Recommender systems survey, Knowl.-Based Syst. 46 (2013) 109–132, http://dx.doi.org/10.1016/j.knosys.2013.03.012.
[2] L. Chen, G. Chen, F. Wang, Recommender systems based on user reviews: The state of the art, User Model. User-Adapt. Interact. 25 (2) (2015) 99–154, http://dx.doi.org/10.1007/s11257-015-9155-5.
[3] R.M. D'Addio, M.A. Domingues, M.G. Manzato, Exploiting feature extraction techniques on users' reviews for movies recommendation, J. Braz. Comput. Soc. 23 (1) (2017) 7, http://dx.doi.org/10.1186/s13173-017-0057-8.
[4] J. McAuley, R. Pandey, J. Leskovec, Inferring networks of substitutable and complementary products, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15, ACM, New York, NY, USA, 2015, pp. 785–794, http://dx.doi.org/10.1145/2783258.2783381.
[5] G. Ganu, Y. Kakodkar, A. Marian, Improving the quality of predictions using textual information in online user reviews, Inf. Syst. 38 (1) (2013) 1–15, http://dx.doi.org/10.1016/j.is.2012.03.001.
[6] M. Terzi, M. Rowe, M.-A. Ferrario, J. Whittle, Text-based user-kNN: Measuring user similarity based on text reviews, in: V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, G.-J. Houben (Eds.), Procceedings of the 22nd International Conference on User Modeling, Adaptation, and Personalization, UMAP 2014, Springer International Publishing, Cham, 2014, pp. 195–206, http://dx.doi.org/10.1007/978-3-319-08786-3_17.
[7] X. Chen, Z. Qin, Y. Zhang, T. Xu, Learning to rank features for recommendation over multiple Categories, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, ACM, New York, NY, USA, 2016, pp. 305–314, http://dx.doi.org/10.1145/2911451.2911549.
[8] M. Capelle, F. Frasincar, M. Moerland, F. Hogenboom, Semantics-based news recommendation, in: Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, ACM, New York, NY, USA, 2012, pp. 27:1–27:9, http://dx.doi.org/10.1145/2254129.2254163.
[9] M. Capelle, M. Moerland, F. Hogenboom, F. Frasincar, D. Vandic, Bing-sf-idf+: A hybrid semantics-driven news recommender, in: Proceedings of the 30th Annual ACM Symposium on Applied Computing, in: SAC '15, ACM, New York, NY, USA, 2015, pp. 732–739, http://dx.doi.org/10.1145/2695664.2695700.
[10] M. de Gemmis, P. Lops, G. Semeraro, A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation, User Mode. User-Adapt. Interact. 17 (3) (2007) 217–255, http://dx.doi.org/10.1007/s11257-006-9023-4.

[11] P. Lops, C. Musto, F. Narducci, M. De Gemmis, P. Basile, G. Semeraro, Mars: A multilanguage recommender system, in: Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '10, ACM, New York, NY, USA, 2010, pp. 24–31, http://dx.doi.org/10.1145/1869446.1869450.

[12] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, Combining distributional semantics and entity linking for context-aware content-based recommendation, in: V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, G.-J. Houben (Eds.), User Modeling, Adaptation, and Personalization: 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7–11, 2014. Proceedings, Springer International Publishing, Cham, 2014, pp. 381–392, http://dx.doi.org/10.1007/978-3-319-08786-3_34.

[13] F. Narducci, P. Basile, C. Musto, P. Lops, A. Caputo, M. de Gemmis, L. Iaquinta, G. Semeraro, Concept-based item representations for a cross-lingual content-based recommendation process, Inform. Sci. 374 (C) (2016) 15–31, http://dx.doi.org/10.1016/j.ins.2016.09.022.

[14] S. Oramas, V.C. Ostuni, T.D. Noia, X. Serra, E.D. Sciascio, Sound and music recommendation with knowledge graphs, ACM Trans. Intell. Syst. Technol. 8 (2) (2016) 21:1–21:21, http://dx.doi.org/10.1145/2926718.

[15] C.D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[16] R. Socher, A. Perelygin, J. Wu, J. Chuang, C.D. Manning, A.Y. Ng, C. Potts, Recursive Deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '13, 2013, pp. 1631–1642.

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

[18] M. Yagci, T. Aytekin, F. Gurgen, On parallelizing SGD for pairwise learning to rank in collaborative filtering recommender systems, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, ACM, New York, NY, USA, 2017, pp. 37–41, http://dx.doi.org/10.1145/3109859.3109906.

[19] D. Liang, J. Altosaar, L. Charlin, D.M. Blei, Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, in: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, ACM, New York, NY, USA, 2016, pp. 59–66, http://dx.doi.org/10.1145/2959100.2959182.

[20] R. Navigli, S.P. Ponzetto, Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network, Artificial Intelligence 193 (2012) 217–250, http://dx.doi.org/10.1016/j.artint.2012.07.001.

[21] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Trans. Assoc. Comput. Linguist. 2 (2014) 231–244, URL http://www.aclweb.org/anthology/Q14-1019.

[22] M.G. Manzato, M.A. Domingues, A.C. Fortes, C.V. Sundermann, R.M. D'Addio, M.S. Conrado, S.O. Rezende, M.G.C. Pimentel, Mining unstructured content for recommender systems: an ensemble approach, Inf. Retr. J. 19 (4) (2016) 378–415, http://dx.doi.org/10.1007/s10791-016-9280-8.

[23] X. Ning, C. Desrosiers, G. Karypis, A comprehensive survey of neighborhood-based recommendation methods, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer US, Boston, MA, 2015, pp. 37–76, http://dx.doi.org/10.1007/978-1-4899-7637-6_2.

[24] C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S.J. Bethard, D. McClosky, The stanford coreNLP natural language processing toolkit, in: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55–60.

[25] R.M. D'Addio, M.G. Manzato, A collaborative filtering approach based on user's reviews, in: Proceedings of the 2014 Brazilian Conference on Intelligent Systems, BRACIS '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 204–209, http://dx.doi.org/10.1109/BRACIS.2014.45.

[26] R.S. Marinho, R.M. D'Addio, M.G. Manzato, Semantic organization of user's reviews applied in recommender systems, in: Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web, WebMedia '17, ACM, New York, NY, USA, 2017, pp. 277–280, http://dx.doi.org/10.1145/3126858.3131600.

[27] A. da Costa Fortes, M.G. Manzato, Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization, in: Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, WebMedia '14, ACM, New York, NY, USA, 2014, pp. 47–54, http://dx.doi.org/10.1145/2664551.2664556.

[28] J. McAuley, C. Targett, Q. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, ACM, New York, NY, USA, 2015, pp. 43–52, http://dx.doi.org/10.1145/2766462.2767755.

[29] F. Wilcoxon, Individual comparisons by ranking methods, in: S. Kotz, N.L. Johnson (Eds.), Breakthroughs in Statistics: Methodology and Distribution, Springer New York, New York, NY, 1992, pp. 196–202.