D'Addio *et al. Journal of the Brazilian Computer Society* (2017) 23:7
DOI 10.1186/s13173-017-0057-8

Journal of the Brazilian Computer Society

RESEARCH Open Access

Exploiting feature extraction techniques on users' reviews for movies recommendation

Rafael M. D'Addio[1]*, Marcos A. Domingues[2] and Marcelo G. Manzato[3]

Abstract
Recommender systems help users to deal with the information overload problem by producing personalized content according to their interests. Beyond the traditional recommender strategies, there is a growing effort to incorporate users' reviews into the recommendation process, since they provide a rich set of information regarding both items' features and users' preferences. This article proposes a recommender system that uses users' reviews to produce items' representations that are based on the overall sentiment toward the items' features. We focus on exploiting the impact that different feature extraction techniques, allied with sentiment analysis, cause in an item attribute-aware neighborhood-based recommender algorithm. We compare four techniques of different granularities (terms and aspects) in two recommendation scenarios (rating prediction and item recommendation) and elect the most promising technique. We also compare our techniques with traditional structured metadata constructions, which are used as the baseline in our experimental evaluation. The results show that the techniques based on terms provide better results, since they produce a larger set of features, hence detailing better the items.

Keywords: Recommender systems, Item representation, Feature extraction, Sentiment analysis

Introduction

Recommender systems emerged to deal with the information overload problem by producing personalized content suggestions to their users. These systems can be traditionally divided into two main strategies: content-based and collaborative filtering [1]. In the content-based approach [2], users' profiles are matched with items' representations using a similarity measure. In collaborative filtering [3, 4], there are two main fields addressed: (i) neighborhood models (also called memory-based models), which find and match ratings from clusters of similar users or items to predict unknown ratings and (ii) latent factor (or model-based) models, which have comprised an alternative path to transform both items and users into the same latent factor space, allowing them to be directly comparable. Beyond these two strategies, there is an effort to combine them into a third hybrid approach, where the flaws of each other are compensated by their strengths [1].

Given the current scenario of the Web, where users can provide content by producing annotations, comments, and reviews about any subject, there is a great amount of rich and detailed information available that is created collaboratively by the community. In spite of its unstructured and uncontrolled nature, user-created descriptions can be exploited by information retrieval and recommender system tasks, lessening the need of domain experts to create structured metadata about the items (indexing). Moreover, one can always obtain updated descriptions about newly added items, which can vary over time depending on the context they are inserted (e.g., news about determined event are bound to vary very fast, while descriptions about movies and books may have little variation through time) [5].

Recent works focus on extending the traditional recommendation paradigms by using this user-provided unstructured information [6–9]. This is a great source of information, since it is able to describe the item and provide feedback about the opinion of the user regarding the item and its features. Indeed, it is common for users to analyze reviews from other users to decide whether or not to consume a certain product. When reading the reviews, a user can verify whether the item meets his/her expectations in certain aspects, analyzing if the majority of the

*Correspondence: rdaddio@icmc.usp.br
[1]Institute of Mathematics and Computer Science, Sao Paulo University, Sao Carlos, SP, Brazil
Full list of author information is available at the end of the article

reviewers consider or not positive the features that he/she thinks interesting.

However, dealing with unstructured text raises a set of challenges, especially when considering user-provided reviews [10]. First, reviews are prone to the occurrence of noise, such as misspelling, false information, and personal opinions that are valid only for the reviewer. Secondly, there is a requirement for natural language processing (NLP) tools to analyze, extract, and structure relevant information about a subject from texts. Finally, there is a lack of research about how to organize and use additional data provided by users in order to enhance items' representations, and consequently, to improve the accuracy of recommendations.

Our proposal focuses on the development of methods that produce items' representations based on users' reviews for recommender systems. For that, we propose an architecture that ranges from text pre-processing to generating recommendations, making use of feature extraction techniques (applied in two granularities: terms and aspects), coupled with sentiment analysis techniques for assigning polarities to the various themes portrayed by users in their reviews. Thus, the items' representations aim to describe items by their characteristics and the collective appreciation of users toward them. Finally, the representations feed a hybrid recommender system that combines a content-based approach with an item-based neighborhood model to produce suggestions to the user.

This article extends previous works [11–13] where two feature extraction techniques (heuristic terms and aspects) were proposed and evaluated using a small dataset. In this article, we propose two new feature extraction techniques: classification terms and hierarchy aspects. The classification terms technique, which was briefly explored in [13], extracts terms by using transductive semi-supervised learning, while the hierarchy aspects technique uses an hierarchical clustering solution to identify topic-related document clusters and elect the most important words of each group to form aspects. With these techniques, we aggregate machine learning into the terms/aspect extraction, differing from the previous techniques, which only relied on simple heuristics.

In addition, we provide a more in-depth evaluation regarding the four techniques proposed by comparing them in two different recommendation scenarios: rating prediction and item recommendation. As part of this work, we still analyze the results in two movie databases that differ greatly in size, being one very small and the other very large. Based on the experiments, we elect and discuss the feature extraction technique for items' representation that performed better among the others. According to our findings, the machine learning techniques produce better results than their heuristic-based counterparts in the majority of the cases.

The article is structured as follows: in "Related work" section we overview some works related to the use of users' textual reviews in the recommendation process; in "The proposed system" section we describe the proposed system. In "Empirical evaluation" section we present our empirical evaluation, detailing the experimental setting, databases and the results. Finally, in "Conclusion and future work" section, we present our conclusions, and discuss current limitations and future work.

Related work

In this section, we present some works related to the use of users' reviews to generate better recommendations for a specific user.

Some recent works use reviews to extract sentiment related to the characteristics of the items in order to characterize them for a content-based recommendation scenario. For example, Qumsiyeh and Ng [8] proposed a system capable of generating recommendations for various multimedia items using information, such as genres, actors, and reviews, extracted from multiple trusted Web sites. Their method is based entirely on mathematical and statistical formulations using the polarity (positive or negative) and degree (ratings, level of appreciation) of every aspect it considers to predict scores of unclassified items. Li et al. [14] proposed a recommender algorithm that captures the contents, such as product information and customer reviews of Web pages related to products and use them to calculate scores and rank these dynamic Web pages. While these works indeed provide item descriptions, they are used mainly in content-based filtering. Our approach, in turn, applies descriptions in a typically collaborative filtering algorithm, addressing content-based major issues such as limited content analysis: in case there are some missing item descriptions, the algorithm can still rely on user ratings to provide recommendation.

Unlike the work described above, other works use reviews for the construction of user profiles, applying them in collaborative filtering and hybrid approaches. Kim et al. [7], for example, proposed a personalized search engine for movies, called MovieMine, based on reviews and user-provided ratings. In this system, the user types a query, which is expanded by adding keywords taken from earlier reviews provided by himself, reflecting his preferences, allowing the search key to be customizable. Wang and Chen [15] proposed a recommendation system that uses reviews to obtain characteristics and their sentiment for the construction of profiles. The recommendation is then performed using these characteristics to find similar users through collaborative filtering and grouping methods. In the system proposed by Aciar et al. [16], text mining techniques are applied to map opinions about items into ontologies that define the skill and knowledge of the user in relation to an item and its features. The recommendation is made through analysis in instances of the proposed ontology. Ganu et al. [6] proposed a review-based recommendation system for restaurants. This system performs a soft clustering of users based on manually predefined topics whose sentiment can be found in the reviews. A great disadvantage of using reviews to produce user profiles is that not every users have the habit of writing reviews, making it impossible to accurately define their preferences. By using those reviews, one can easily describe items better, since the information on that end is more abundant, i.e., there are more users describing an item than a user describing several items. Moreover, one can find additional reviews and descriptions about items in other sources on the Web.

In a third approach, recent works make use of users' reviews to derive ratings, which will be used as a complement to the ratings already assigned by the user in the collaborative filtering process. For example, in the work of Ganu et al. [6], described above, text-based ratings were also produced, and their values were compared to the traditional ratings using various recommendation algorithms, as neighborhood and latent factors-based models. Finally, both ratings were compared with the soft clustering model proposed. Pero and Horváth [9] proposed a framework that uses sentiment analysis to produce text-based ratings which are processed in conjunction with traditional ratings on a matrix factorization algorithm. The main drawback of this approach is that by considering only a single rating for the whole review, the system may loose information about the preferences of the user and/or the quality of the item in regard to different aspects.

This work differs from the aforementioned since it uses reviews to produce item representations that will feed an algorithm typically used in collaborative filtering scenarios. Some of the related works previously reported, such as the works of Ganu et al. [6] and Aciar et al. [16], use manually predefined sets of features. Our work, in turn, focuses on feature extraction and selection methods that will automatically detect relevant features from the reviews. This allows the applicability of our approaches into different product domains, positively affecting the generalization of our approaches. We also provide a thorough study, by comparing the impact of different techniques, based on heuristics and machine learning, on the recommendation accuracy.

The proposed system

As previously stated, the proposed system uses users' reviews to produce representations about the items, containing their most relevant characteristics and the overall sentiment toward them. Thus, the preference of a user is built based on an average of opinions.

For example, in the reviews about the movies that a user u rated positively it was reported that they have strong science fiction elements (positive sentiment), but have weak or absent (negative or neutral/absent sentiment) romance. Another user v has evaluated positively movies that have direction and photography praised in their reviews and evaluated negatively movies that contain positive critics about their suspense. Thus, the system will most likely suggest movies that relates to the preferences of those users: for user u, it will suggest science fiction movies with no aspects of romance, while for user v the suggestions will have positive direction and photography, and negative or absent suspense.

In order to do that, we developed a system's architecture with concise and specific modules. As depicted in Fig. 1, the main goal of the system is to produce a set of items' representations that will feed, alongside the ratings provided by the users, the recommendation algorithm.

Firstly, the reviews go through a pre-processing step (detailed in the "Text pre-processing" section) in order to reduce noise and provide a structured version of them. Next, the feature extraction module obtains relevant characteristics about the domain of the items. This module is the main challenge addressed in this work, and four different techniques were tested, being two of them based on terms and the other two based on aspects. Also, for each granularity (terms and aspects), we explore one technique based on heuristics and one based on machine learning and verify which ones provide the best results. This module will be detailed in the "Feature extraction" section. In the item's representation creation module, the features extracted previously are represented as the positions of each item's vector, and their scores are computed as the overall sentiment toward them obtained in the item's reviews. This step, which is detailed in the "Sentiment analysis and item representations construction" section, is performed for each item present in the database, producing a set of items' representations. These representations feed a neighborhood-based collaborative filtering algorithm, which is detailed in the "Recommendation" section.

Text pre-processing

The pre-processing module is responsible for carrying out several routines that aim to structuring the reviews. We apply, therefore, the following routines:

- *Noise removal*: we remove stop-words, dates, numeric characters, Web pages, and words with special characters;
- *Tokenization, stemming, and lemmatization*: we divide each word into a token and extract its stem and lemma. The stem and lemma are used according to the feature extraction technique in the feature extraction module;

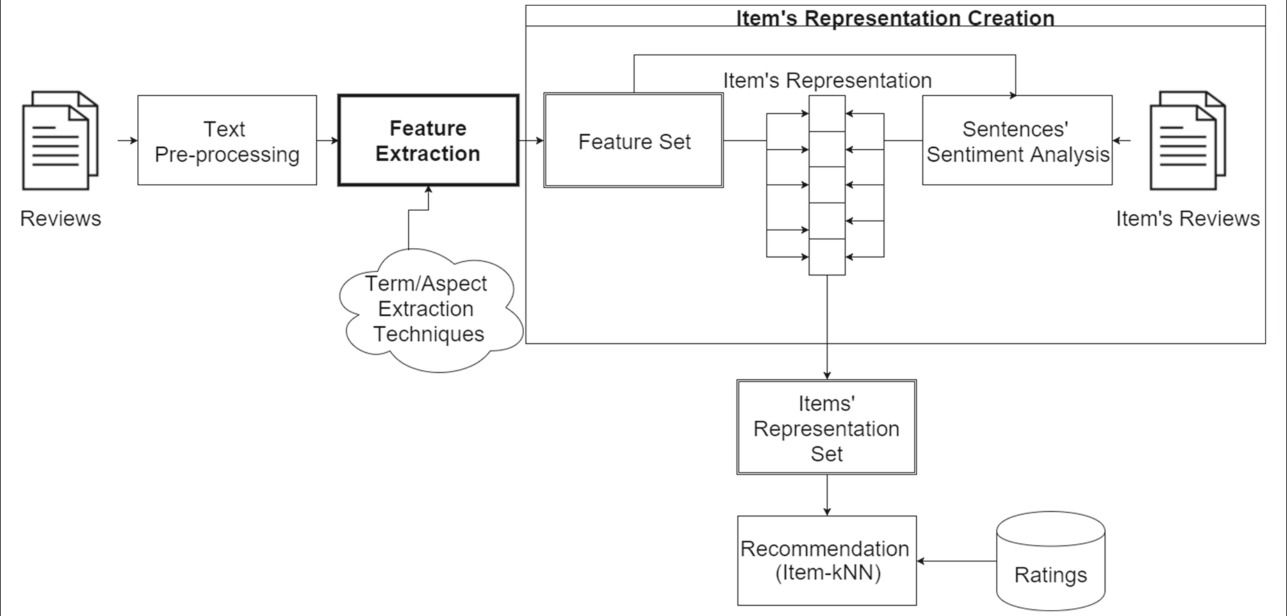

Item's Representation Creation
Item's Representation
Reviews
Text Pre-processing
Feature Extraction
Feature Set
Sentences' Sentiment Analysis
Item's Reviews
Term/Aspect Extraction Techniques
Items' Representation Set
Recommendation (Item-kNN)
Ratings

Fig. 1 The proposed system's architecture

- *Part-of-speech tagging (POS tagging)*: we obtain the POS tag of each token. This feature will be used in all feature extraction techniques;
- *Sentence splitting*: we split the text into sentences and divide those that are compound so they can be processed by the sentiment analysis algorithm;
- *Parsing*: we apply a parser in the texts, so that more sensitive information is obtained, such as the structure of sentences and relationships between words. Such information will be used by some of the techniques in the feature extraction module.

In our work, these routines are supported by the well-known Stanford CoreNLP[1] [17], a natural language processing toolkit that contains several NLP routines. The CoreNLP also supports a sentence-level sentiment analysis algorithm [18], which we also apply during this preprocessing step. More details about this algorithm can be found in the "Sentiment analysis algorithm" section. Finally, the stemming was performed through the traditional Porter algorithm [19].

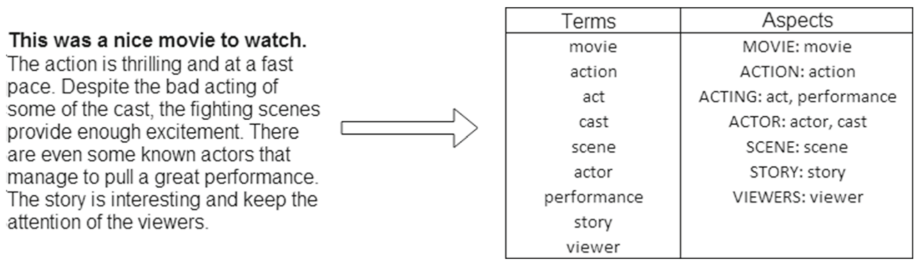
The resulting output of this procedure are Extensible Markup Language (XML) documents containing the texts in a more structured form. In those files, each of the documents is described in a tree structure that segments them into sentences, in which each one of them may contain its syntactic parsing, a sentiment and the tokens. Each token, in its turn, contains its lemmatized and stemmed form and its POS tag.

Feature extraction

The feature extraction module is responsible for extracting features from the texts that will compose the item vectors of the system.

The most common way to represent characteristics of a text is through terms. Terms are words extracted from a text describing the subjects covered in it [20]. Especially in product reviews, terms are predominantly nouns, due to the fact that these reviews refer to nominal characteristics of the products, for example: "the **resolution** of this **camera** is good" or "this **actor** was not convincing". In the sentiment analysis area, in turn, characteristics tend to be represented as aspects, which are collections of terms that represent the same topic [21]. Figure 2 illustrates the difference between terms and aspects. In the figure, the review was processed and only the most important words were maintained as terms (first column). From them, we could produce aspects, which are depicted in the second column and may contain more than one term. For instance, the aspect "ACTING" contains the terms "act" and "performance".

In this work, we compare the effect that these two representations causes to a recommendation algorithm that uses information about items. Therefore, we apply two term extraction and two aspects extraction techniques in the set of reviews, and the result of each is evaluated on the recommender system. Our evaluation also presents a discussion about which technique performed better in the particular domain of movies



This was a nice movie to watch.
The action is thrilling and at a fast pace. Despite the bad acting of some of the cast, the fighting scenes provide enough excitement. There are even some known actors that manage to pull a great performance. The story is interesting and keep the attention of the viewers.

Terms	Aspects
movie	MOVIE: movie
action	ACTION: action
act	ACTING: act, performance
cast	ACTOR: actor, cast
scene	SCENE: scene
actor	STORY: story
performance	VIEWERS: viewer
story	
viewer	

Fig. 2 Terms and aspects extracted from a review sample

recommendation. In the following sections, we introduce each of these techniques.

Extracting terms through heuristics

The first term extraction technique involves the application of two filters in the set of lemmatized words: one linguistic and another statistical.

First, we select only words with the noun POS tag as candidate terms. One of the problems with part-of-speech taggers is that unknown words tend to be classified as nouns. This problem is aggravated when using texts produced by users, due to misspellings, Internet slangs and abbreviations.

Therefore, we select from the set of candidate words those that are more common among the item reviews, assuming that these may be, in fact, features. Since an item has n reviews, instead of using the document frequency (DF) [20], we decided to use a similar metric called item frequency (IF) [11, 12]. Considering F as the candidate words set and I the items set, the item frequency IF_f of a candidate word f is given by

$$IF_f = \sum_{i}^{|I|} k_{if}, \qquad (1)$$

where k_{if} is equal to 1 if an item i has the candidate word in at least one of its reviews. The IF_f is then compared to a threshold, and if its value is greater than it, the candidate word is maintained in the term set. In an earlier experiment, we considered four different thresholds for constructing lists of terms: 1, 30, 100, and 200 [12]. By considering these threshold values, we verified the impact of different sizes of term sets, as well as what types of terms should be regarded: those more specific, i.e., that may appear in at least two items (threshold 1) or those more general, i.e., that may appear in a larger set of items (threshold 200). The results indicated that the threshold of 30 is a good value since it provides the best trade-off between term set size and recommendation accuracy, performing better than the baseline.

Extracting aspects through heuristics

In this approach, we apply a set of heuristics similar to those applied in the term extraction, to reduce the number of candidate words and create aspects from the reviews. We apply in the set of lemmatized word both linguistic and statistical filters applied in the term extraction technique: first, we select only nouns and then apply the item frequency, discarding the candidate words that have an IF_f value lower than a certain threshold. Based on the results obtained in the previous approach, we apply the threshold of 30 since the terms that appear in a number of documents smaller than 30 do not heavily affect the results and thus can be regarded as noise.

From the set of remaining terms, we group those that contain the same or similar stem. For example, when performing the pre-processing step, we may obtain the lemmas "director" and "direction," but both share the same stem "direct." By grouping the lemmas that share the same stem, we often reduce the number of features that have relation to the same topic.

The last step, performed semi-automatically, is to group synonymous topics. We use the WordNet[2] lexical database as a basis to obtain the synonyms of the lemmas for each existing topic and group those topics who share the synonyms. After performing this step, we make a manual check to remove errors and noise.

We explored this technique in a previous experiment [11]. The results showed that the produced aspect set was very small, which affected the capability of the recommender to distinguish the items and hence to locate appropriate neighbors to produce adequate suggestions.

Extracting terms through transductive learning

The second term extraction technique is called TLATE (transductive learning for automatic term extraction) and was proposed by Conrado et al. [22]. This method uses transductive semi-supervised learning to classify if a word is a term or not. It is executed in three steps in addition to the text pre-processing already performed: (i) word feature extraction, (ii) filtering, and (iii) transductive classification.

In the word feature extraction step, we extract information that aims to characterize each word using features that range from simple statistical (such as DF, for instance) and linguistic (POS tags, for instance) knowledge to hybrid measures that utilize both kinds of knowledge. In total, there are 24 features which can be seen with more details in [22]. With this, each word is represented by a vector where each position represents one of the aforementioned measures.

The filtering step aims to remove words that have less chance to be terms. In a previous work, we tested two different filters [13]: (1) filter_DF, which removes the words that occur only in one document in the database and (2) filter_DF_N that also deletes those words that are not nouns. In our experiments, we found out that the filter_DF_N performed better since it provides a significantly smaller set of candidate words to be classified by the transductive learning step, and hence the overall outcome contains a smaller but more descriptive terms set.

In the transductive learning step, we represent the candidate words set in a mutual k nearest neighbors (k-NN) network [23] and test $k = \{7, 57\}$, since these are values previously used by Conrado et al. [22] and cover two very different settings: one with a limited number of neighbors and one with a larger number. To calculate the similarity between word vectors, we use the Euclidean distance, since they are non-sparse numeric vectors. This network is used by a transductive learning algorithm that classifies the words into terms or non-terms. The label spreading of words was performed using the LLGC (learning with local and global consistency) algorithm [24] with the regularization parameter (μ) set as 0.9, defined in preliminary experimentation. Since transductive classification is capable to learn from smaller training sets [22], we selected 16 of the word vectors that comprehend the basic characteristics of a movie (genres, direction, photography, among others) and labeled them as terms. We also select 16 unrelated word vectors (for instance, words that can have both adjective and noun POS) as non-terms. LLGC learns from the information of these labeled words, and the remaining unlabeled words are used to perform the classification. In our analysis, the LLGC produced better sets of terms with the k of the k-NN network set as 57.

Extracting aspects through hierarchy clustering

The second aspect extraction technique was proposed in order to eliminate the human intervention required in the previous technique. We use the LIHC (LUPI-based incremental hierarchical clustering) [25] for the automatic generation of a hierarchical clustering of texts, and through this, produce a topic hierarchy. This topic hierarchy is then processed so that the most representative topics are used as aspects of the items.

The LIHC makes use of technical and privileged information to perform the document grouping task. The technical information used is a traditional bag-of-words representation, containing the frequency of the terms present in the document. Privileged information in text processing domain consists of information besides traditional term frequency (TF) or term frequency-inverse document frequency (TF-IDF) [25]. In this work, we use the part-of-speech tag of words as privileged information, since they represent a linguistic information about the terms located in the documents.

This method considers that each document is represented by two vectors located in their respective spaces: technical and privileged. We utilize a consensus-based clustering method that analyzes several clusters produced by different algorithms or the same algorithm with different parameters and combines them into a single clustering model. For our proposal, we use the well-known k-means algorithm [26], applying it repeatedly with 50 different parameter settings (i.e., varying k from 1 to 50), into both documents representations.

The clusters produced are combined using a co-association matrix M. Thus, for each type of information, we generate its respective co-association matrix: M and M^*. These matrices, in turn, are combined into a final co-association matrix M^C.

Finally, we apply the classical hierarchical clustering algorithm UPGMA (unweighted pair group method with arithmetic mean) [27] in the matrix M^C, producing a hierarchical clustering model of documents.

We use the hierarchical model produced to extract topics that represent the aspects of the items, similarly to the work performed by Domingues et al. [28]. We use the F1-measure to extract the five most important words of every (sub)cluster in the hierarchy in the following manner: each (sub)cluster contains a set of D documents and a set of T candidate terms, present within the documents. Each word in the set T is capable of retrieving a set of documents D_{term} from the whole dataset, which we compute the F1 measure against D. The terms that contain the highest F1 scores are considered the most important terms since they are able to retrieve a D_{term} set that are very similar to D. With this, each set of words corresponds to an aspect. Next, we select a determined set of aspects by selecting (sub)clusters with size within the range of the minimum and maximum number of documents, defined by experimentation. We select topics with the following value ranges: $[2, 7]$, $[5, 10]$, $[5, 100]$, $[10, 15]$, $[10, 50]$, $[15, 20]$, and $[50, 100]$, where the first value corresponds to the minimum number of documents and the second value corresponds to the maximum number of documents. With these values, we explored different cluster granularities, which, in turn, allowed us to observe the specificity of the aspects; those produced in (sub)clusters with $[2, 7]$ documents are more specific, while those produced in (sub)clusters with $[50, 100]$ documents are more generic. By experimentation, the granularity $[2, 7]$ provided the best results, as it contains more specific descriptions about the items. This configuration is used in the experimental evaluation presented in the "Empirical evaluation" section.

Sentiment analysis and item representations construction

At the end of the feature extraction module, the resulting set is used by the items' representations generation module. In this module, the sentiment value for each item's feature is computed. Thus, an item is represented by the average sentiment of many users' reviews toward each of its characteristics.

This module is divided into two steps. First, we apply a sentiment analysis algorithm in the item's reviews, obtaining the sentiment for each sentence. The main reason for using a sentence-level sentiment analysis is that most of the features extracted from the reviews are nouns, especially in a movie recommendation domain. Nouns have neutral sentiment; hence, we rely on the context and sentiment existing in sentences containing these nouns. In the second step, for each feature, we select all sentences that relate to it and calculate its average sentiment. These steps will be better detailed in the following subsections.

Sentiment analysis algorithm

As stated earlier, we perform in the pre-processing step a sentence splitting in the reviews, so they can be processed by the sentiment analysis algorithm, resulting in a set of sentiment information of all reviews' sentences. In the adopted approach [18], recursive neural networks models are used to build representations that capture the structure of the sentences, obtaining in this way their sentiment based on the meaning of each word.

The algorithm splits the tokens of a sentence and calculates the sentiment by combining the tokens and constructing a tree in a bottom-up approach, where the root node is the final sentiment for the whole sentence and may contain one of the 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."

Item's vector construction

In the final step of the item's representation creation, the system analyzes the feature set and checks whether they are terms or aspects. If they are terms, the system verifies whether they are represented as a stem or lemma and converts them into a set of words that the stems/lemmas comprehend. Then, for each item, the system looks up in the XML files containing its pre-processed reviews which sentences contain these words. If the feature set are aspects, the system finds the set of terms that the aspect represents, and then applies the same stem/lemma recognition for each term, finally converting them into words and checking the sentences which contain them.

After having obtained the sentences related to each feature, the next step is the sentiment attribution to them. For each feature of each item, it is calculated as the average sentiment of the related sentences. Thus, the final value represents the collective level of appreciation or depreciation of a certain attribute of an item. As mentioned before, the sentiment values range between 1 and 5, being 1 equals to "Very Negative" while 5 means "Very Positive." A zero value indicates that an item simply does not have that feature.

As an alternative for the vector construction, the system can also produce a binary sentiment vector. In a previous work, we defined the best strategy to perform the binarization of the sentiment vectors [12], which we describe as follows: each feature is divided into two portions, corresponding to its positive and negative parts. Features that have positive sentiment are attributed value 1 to its positive portion, while those with negative sentiment have their negative portion defined as 1. We used a threshold α to adjust the relevance of the intensity of the sentiment in the binarization process, and through experimentation, it was determined that tighter intervals produced the best results. With the aspect-based approach described in the "Extracting aspects through heuristics" section, the recommender performed better with the binary representations. We argue that since the binary scoring duplicates the number of features, the aspect-based approach has bigger vectors to describe the items, thus producing better results.

To illustrate this step, Fig. 3 presents the sentiment assignment to the aspect "acting," extracted from the review sample in Fig. 2. Considering that an item i in the system has only the review in Fig. 2, the system finds the sentences that make reference to "acting," obtains their polarities, and calculates the average sentiment. Finally, if necessary, the system performs the binarization step.

Recommendation

In the recommendation module, the items' representations are finally analyzed alongside the ratings provided by the users. We use a neighborhood-based recommendation algorithm because of its simplicity, efficiency, stability, as well as the easiness for extending the traditional model into using items' feature vectors [3]. We opted to use an item k-NN collaborative filtering algorithm that takes into account the bias of users and items [29], and this one was adjusted to use the items' feature vectors in the process of obtaining the neighbors instead of the traditional users' ratings vector. With that, our approach can be regarded as a hybrid recommender, since it uses content-based

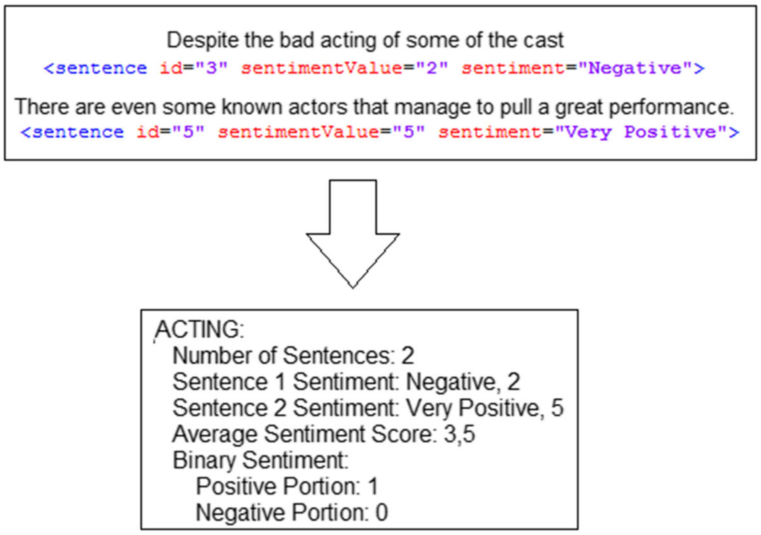

Despite the bad acting of some of the cast
<sentence id="3" sentimentValue="2" sentiment="Negative">
There are even some known actors that manage to pull a great performance.
<sentence id="5" sentimentValue="5" sentiment="Very Positive">
ACTING:
Number of Sentences: 2
Sentence 1 Sentiment: Negative, 2
Sentence 2 Sentiment: Very Positive, 5
Average Sentiment Score: 3,5
Binary Sentiment:
Positive Portion: 1
Negative Portion: 0

Fig. 3 Example of calculating the average sentiment score for the aspect "ACTING"

representations in an item-based collaborative filtering algorithm.

A common approach of collaborative filtering algorithms is to adjust the data for accounting item and user bias. These effects are mainly tendencies of users to rate items in different manners (higher or lower ratings) or items that tend to be rated differently than the others. We encapsulate these effects within the baseline estimates. A baseline estimate for an unknown rating $\hat{r}_{ui}$ is denoted by:

$$b_{ui} = \mu + b_u + b_i, \tag{2}$$

where μ is the global average rating and b_i and b_u are the item's and user's deviations from the average. To estimate b_u and b_i, one can solve a least squares problem. We adopted a simple approach which will iterate a number of times the following equations:

$$b_i = \frac{\sum_{u:(u,i)\in K}(r_{ui} - \mu - b_u)}{\lambda_1 + |\{u|(u,i) \in K\}|}, \tag{3}$$

$$b_u = \frac{\sum_{i:(u,i)\in K}(r_{ui} - \mu - b_i)}{\lambda_2 + |\{i|(u,i) \in K\}|}, \tag{4}$$

where K is the set of rated items and r_{ui} is a rating given by a user u to an item i. In our experiments, we iterated 10 times these equations following the order in which they appear and defined through preliminary experimentation the values of 10 and 15 for the constants λ_1 and λ_2, respectively.

The goal of the recommendation algorithm is to find similar items rated by a user and to predict a rating based on the ratings of those similar items. In this way, a rating is predicted for an unobserved user-item pair by considering the similar items he/she already rated.

In order to find similar items, a similarity measure is employed between the previously described items' representations. This similarity can be based on several correlation or distance metrics, and through experimentation, we found out that the Pearson correlation coefficient, p_{ij}, performed better in most of the settings. In this metric, a value corresponding to 1 means total correlation, in which the vectors coincide and are in the same direction, whereas a value of -1 corresponds to the case where the vectors coincide but are in opposite directions. The Pearson correlation is defined as follows:

$$p_{ij} = \frac{\sum_{n=1}^{k}\left(w_n^i - \overline{w_i}\right)\left(w_n^j - \overline{w_j}\right)}{\sqrt{\sum_{n=1}^{k}\left(w_n^i - \overline{w_i}\right)^2}\sqrt{\sum_{n=1}^{k}\left(w_n^j - \overline{w_j}\right)^2}}, \tag{5}$$

where $\overline{w_i}$ and $\overline{w_j}$ correspond to the average value of the features of i and j, respectively. The final similarity measure is a shrunk correlation coefficient, s_{ij}:

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_3} p_{ij}, \tag{6}$$

where n_{ij} is the number of features that describe both items i and j, and λ_3 is a regularization constant, set as 100 since this value penalizes the similarity of items who have fewer features in common [29].

Given the items' feature-based representations and the similarity measure presented previously, we identify the k items rated by u that are most similar to i, the k-nearest neighbors. Similarly to Koren [29], we denote this set as $S^k(i;u)$. Using this set, the final predicted rating is an average of the k most similar items' ratings, adjusted to their baseline estimates:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S^k(i;u)} s_{ij} \left(r_{uj} - b_{uj} \right)}{\sum_{j \in S^k(i;u)} s_{ij}}. \quad (7)$$

In our experiments, we used $k = \{20, 40, 60, 80, 100\}$. This way, we can check the effect of each type of item representation in different sizes of neighbors.

Empirical evaluation

To evaluate our proposal, we selected the most promising configuration of each of the aforementioned feature extraction techniques and compare them with each other as well as with some traditional structured metadata representations.

All experiments were carried out in a 10-fold cross-validation setting. The rating database was randomly divided into 10-folds, where each fold contained at least one interaction of each user. This way, we guarantee that the system can generate recommendations for every user in the database, and that every user can be inserted at the same time in the test set. 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. Since the reviews used are collected from outer sources, they can be viewed as additional content and thus the representations are not divided into 10-folds.

The values displayed as results are the average values of the iterations. In order to check the significance of the results, we applied the Student's t test [26].

Databases

We performed our experiments on two databases related to movies, generated from the MovieLens Web site[3] and enhanced with information contained in the Internet Movie Database (IMDb) Web site[4].

In preliminary experiments [11–13], we used only the well-known MovieLens 100k (ML-100k) database due to the fact that it is smaller and therefore simpler to find additional information about items, as well as being quicker to perform the experiments. The ML-100k consists of 100,000 ratings (from 1 to 5) performed by 943 users for 1682 movies.

In this article, we extend our findings by adopting the HetRec2011 Movielens-2k (HetRec ML) database[5]. It is a significantly larger dataset, consisting of 855,598 ratings (from 1 to 5) performed by 2113 users for 10,197 movies.

We have enhanced both databases by collecting structured metadata such as genres, actors, and directors from the IMDb Web site. We constructed binary items' representations for each of these metadata, where 1 means that the item has the metadatum while 0 means it does not, and used them as baseline representations, applying them to calculate the item similarity by the same means that our approaches are used. In our evaluation, we compare the descriptive power of our representations against those binary, baseline representations. Table 1 shows the total and average number of features for each structured metadatum considered.

We also collected up to 10 reviews per item for the ML-100k database, resulting in a total of 15,863 documents and up to 100 reviews per item for the HetRec database, resulting in a total of 656,031 documents. The reviews were selected as the top-N from IMDb, ordered by their helpfulness. Unfortunately, not every movie had the maximum number of reviews, in fact, there were some movies that did not have reviews at all. The reason we selected only 10 reviews for the ML-100k database is that this database was used in our preliminary experiments; hence, we needed a smaller set of reviews to guarantee speed in these experiments. With the preliminary experiments done, we were able to test the best configurations in a larger setting, which demands more time.

Evaluation metrics

In order to evaluate which technique describes better the items for a recommender systems, the resulting recommendations were evaluated in two main scenarios: rating prediction and item recommendation [30].

Rating prediction evaluates how much the ratings predicted by the system deviate from real ratings assigned by users. For that, we used the root mean square error (RMSE) metric, which is defined as:

$$\text{RMSE} = \frac{1}{|U|} \sum_{u \in U} \sqrt{\frac{1}{|O_u|} \sum_{i \in O_u} \left(\hat{r}_{ui} - r_{ui} \right)^2}, \quad (8)$$

where O_u is the predicted items set that the user u evaluated, $\hat{r}_{ui}$ is the predicted rating, and r_{ui} is the real rating.

Item recommendation evaluates the capacity of the system to generate personalized rankings of suggestions. We evaluated this scenario by selecting the top 100 items with the highest predicted ratings for each user and applied the precision at n (prec@n) and the mean average precision (MAP) measures. The prec@n measures how many relevant items are returned in relation to a small n sample of the total ranking:

Table 1 Total and average values for each structured metadata for ML-100k and HetRec2011-ML2k databases

		Total	Average occurrence
ML-100k	Actors	44,178	44.15
	Directors	1016	1.06
	Genres	18	1.72
HetRec ML	Actors	95,321	22.78
	Directors	4060	1.0
	Genres	20	2.04

$$prec@n = \frac{\#(\text{relevant items in } n)}{n}. \quad (9)$$

We use this measure with $n = 10$. The MAP measure, in its turn, evaluates the whole ranking, but gives 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 [20]:

$$MAP = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} prec@R_{jk}. \quad (10)$$

Feature extraction techniques comparison

In this section, we present a comparative study about the applicability of the four aforementioned feature extraction techniques in a recommender system. For each technique, we selected the best configuration and compared them with each other and to structured metadata. We named the four techniques as:

- *Heuristic terms*: the term extraction technique described in the "Extracting terms through heuristics" section, using $IF = 30$;
- *Classification terms*: the term extraction technique described in the "Extracting terms through transductive learning" section, using the filter_DF_N in TLATE's filtering step and the k-NN network with $k = 57$ in the transductive step;
- *Heuristic aspects*: the aspect extraction technique described in the "Extracting aspects through heuristics" section, using the binarized sentiment approach in the items' representation creation module;
- *Hierarchy aspects*: the aspect extraction technique described in the "Extracting aspects through hierarchy clustering" section, using the topic granularity of $[2, 7]$.

Table 2 shows the total number of features for each technique, as well as the average number of features that the items contain.

In the following subsections, we present the results of the comparison for each recommendation scenario separately. Finally, we discuss our findings.

Rating prediction

In this section, we present the results obtained in the rating prediction scenario. Table 3 shows the results for the structured metadata baseline for both databases, while Table 4 presents the results related to our proposed approaches.

Table 2 Total and average values of each feature extraction approach selected for the ML-100k and HetRec2011-ML2k databases

		Total	Average occurrence
ML-100k	Heuristic terms	3085	223.32
	Classification terms	8433	401.07
	Heuristic aspects	78	22.59
	Hierarchy aspects	933	236.65
HetRec ML	Heuristic terms	33,618	840.89
	Classification terms	17,864	1469.43
	Heuristic aspects	55	41.59
	Hierarchy aspects	5428	2097.93

Figure 4 presents a graphic comparing the baselines and the approaches for the ML-100k database, and Fig. 5 shows the comparison in the HetRec ML database.

In relation to the ML-100k database, we observed that the term-based approaches produced better results than those produced by the aspect-based approaches. By applying a Student's t test, we observed that, considering smaller neighbors values ($k = \{20, 40, 60\}$), both term-based techniques produced statistically better results than the others, using a p value <0.005. As it can be seen, both techniques based on machine learning presented better results than their heuristic counterpart. The hierarchy aspects, even though performing similarly to the actor metadata baseline, is statistically superior than the heuristic aspects approach. The classification terms approach presents better results than the heuristic approach at $k = \{20, 40, 60\}$, but these values are not statistically significant.

In the HetRec ML database, it can also be seen that term-based approaches have the best results, being statistically better than the others, using a p value <0.001. In this dataset, in turn, the classification terms approach performed statistically better than the heuristic terms approach, also considering a value of $p < 0.001$. The hierarchy aspects approach also showed significant results,

Table 3 The average RMSE results for the structured metadata baseline in both databases

		$k = 20$	$k = 40$	$k = 60$	$k = 80$	$k = 100$
ML 100k	Actors	0.9384	0.9383	0.9382	0.9382	0.9381
	Directors	0.9438	0.9438	0.9438	0.9438	0.9438
	Genres	0.9404	0.9401	0.9402	0.9401	0.9401
HetRec ML	Actors	0.8229	0.8226	0.8225	0.8225	0.8224
	Directors	0.8311	0.8311	0.8311	0.8311	0.8311
	Genres	0.8294	0.8283	0.8280	0.8280	0.8280

Table 4 The average RMSE results for the feature extraction techniques applied in both databases

		$k = 20$	$k = 40$	$k = 60$	$k = 80$	$k = 100$
ML 100k	Heuristic terms	0.9310	0.9314	0.9330	0.9347	0.9361
	Classification terms	0.9302	0.9306	0.9328	0.9345	0.9360
	Heuristic aspects	0.9424	0.9409	0.9403	0.9403	0.9404
	Hierarchy aspects	0.9406	0.9383	0.9381	0.9384	0.9388
HetRec ML	Heuristic terms	0.8025	0.8030	0.8050	0.8071	0.8090
	Classification terms	0.7964	0.7978	0.8005	0.8030	0.8052
	Heuristic aspects	0.8289	0.8267	0.8270	0.8277	0.8285
	Hierarchy aspects	0.8173	0.8168	0.8184	0.8200	0.8214

being statistically better than the baseline results and the heuristic aspects approach, for a value of p value <0.001, when considering a small number of neighbors ($k = \{20, 40, 60\}$).

Item recommendation

In this section, we present the results related to the item recommendation scenario. Table 5 exhibits the baseline results for both databases, while Table 6 presents the results for the proposed approaches.

Figure 6 presents graphics with the results of prec@10 and MAP of the baselines and the proposed approaches for the ML-100k database, and Fig. 7 shows the comparison in the HetRec ML database.

Regarding the results obtained in the ML-100k database, it can be seen that for the prec@10 the results of all approaches are statistically better than those produced by the baselines for a value of $p < 0.005$. The MAP results, in turn, are statistically better for the same p value, except for those produced by the heuristic aspects approach. Again, we noted that term-based approaches produce better results than others, but there are no statistically difference between them. One can also see that both machine learning approaches provide better results, especially considering the hierarchy aspects in relation to heuristic aspects, whose results are statistically different.

For the HetRec ML database, we observed that for prec@10, except heuristic aspects, all approaches produce better results than those produced by the baselines, considering larger numbers of neighbors ($k = \{60, 80, 100\}$), but only the term-based approaches have statistically superior results, with p value <0.005. The results obtained for MAP, in turn, were not relevant, being lower or closer to those produced by structured metadata.

Discussion

As it can be observed, term-based approaches produced better results in both databases and recommendation scenarios. This implies that a greater number of features tend to describe better the items than a significantly smaller set, considering the recommendation algorithm used in this work. While the term-based approaches contained thousands of features, aspect-based approaches contained on average a few hundred due to their nature of grouping topic-related words.

In the term-based approaches, considering the rating prediction scenario, it can be seen that as the number of neighbors increases, the accuracy of the predictions decreases. One possible reason for this effect is that as the items are described in greater detail, there is a possibility that as soon as one increases the number of neighbors, unrelated items will be included in the prediction calculation. The opposite effect, however, occurs in the aspect-based approaches. It can be observed that if the number is increased to a certain number of neighbors, the prediction accuracy improves. It is speculated that this is due to the fact that the representations contain a very small number of features, making it difficult to tell them apart, and hence, it needs more neighbors to calculate the rating prediction.

Regarding the results for the item recommendation scenario, we observed that those produced by all approaches

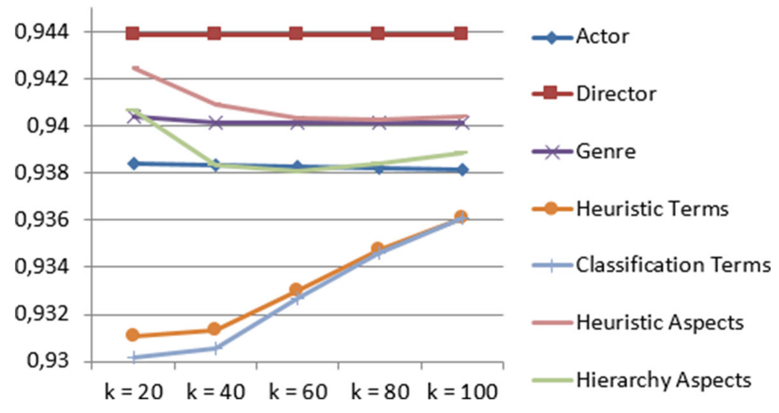


Fig. 4 Graphics showing the RMSE results for the ML-100k database

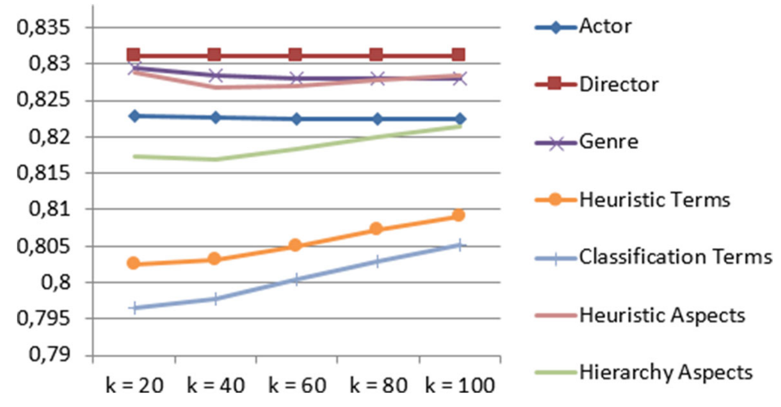


Fig. 5 Graphics showing the RMSE results for the HetRec ML database

outperform the other results, on average, in 1% on prec@10, while for the MAP there are no conclusive results. Although the results for the ML-100k database are favorable, those obtained for the HetRec ML do not exceed those obtained by the baselines. Despite that the prec@10 results are favorable, there is not a significant contribution to the item recommendation scenario. This is due mainly to the nature of the implemented algorithm. Since it performs the rating prediction task and then through the ratings, it generates a ranking of the 100 items with the highest predicted rating, it is assumed that the final ranking contains items which are not in the test set, since it is a drawn portion of the 10% of the database. Since the goal of this article was to evaluate the impact of the feature extraction techniques, and for this, we chose a *k*-NN algorithm initially developed for rating prediction tasks, we leave for future work the extension of this model to produce better rankings.

As observed in the results, in general, the machine learning approaches provided better results than their heuristic counterpart. This can be better observed in relation to the aspect approaches, where the results are statistically different in every metric and database used in this study. In relation to the term approach, which provided the best results, although the classification approach was not statistically superior to the heuristic approach for the ML-100k database, the experiment carried out in the HetRec ML database indicates that the usage of the TLATE can produce more accurate representations with a larger set of reviews.

To further analyze the differences among our approaches, Table 7 presents the top 10 most frequent features extracted with each technique in both datasets used in our experiments. As explained before, some techniques use the lemmatized version of words, while others use the stemmed version. We maintain this format here.

As it can be seen, both top 10 lists generated from the term extraction techniques are almost equivalent, with differences on the order or with few words. In fact, the first three words are the same in each technique, and they are indeed relevant: the first two regard the opinion of users toward the item as a whole, while the third regards the time, or duration, of a movie. This is also reflected in the heuristic aspect technique, where both words "film" and "movie" are part of the "cinematography" aspect, which is the most frequent in both datasets. Another difference worth noting is that classification terms removed words such as "way" and "thing," which are very common nouns but can be considered of little impact on describing movies. Lastly, as it can be seen, hierarchy aspects produces not semantically

Table 5 The average values of prec@10 and MAP for the structured metadata baseline applied in both databases

		$k = 20$		$k = 40$		$k = 60$		$k = 80$		$k = 100$	
		prec@10	MAP	prec@10	MAP	prec@10	MAP	prec@10	MAP	prec@10	MAP
ML-100k	Actors	0.0892	0.0579	0.0893	0.0579	0.0894	0.0579	0.0892	0.0580	0.0892	0.0580
	Directors	0.0872	0.0571	0.0872	0.0571	0.0872	0.0571	0.0872	0.0571	0.0872	0.0571
	Genres	0.0843	0.0568	0.0849	0.0570	0.0849	0.0570	0.0849	0.0570	0.0849	0.0570
HetRec ML	Actors	0.1081	0.0238	0.1073	0.0255	0.1073	0.0255	0.1073	0.0255	0.1090	0.0255
	Directors	0.1047	0.0270	0.1047	0.0270	0.1047	0.0270	0.1047	0.0270	0.1047	0.0270
	Genres	0.1021	0.0280	0.1030	0.0280	0.1038	0.0281	0.1038	0.0281	0.1038	0.0281

Table 6 The average prec@10 and MAP results for the feature extraction techniques applied in both databases

		k = 20		*k* = 40		*k* = 60		*k* = 80		*k* = 100	
		prec@10	MAP	prec@10	MAP	prec@10	MAP	prec@10	MAP	prec@10	MAP
ML-100k	Heuristic terms	0.1041	0.0656	0.1059	0.0671	0.1021	0.0673	0.1039	0.0675	0.1024	0.0676
	Classification terms	0.1043	0.0658	0.1051	0.0671	0.1044	0.0675	0.1048	0.0676	0.1042	0.0677
	Heuristic aspects	0.0951	0.0597	0.0956	0.0604	0.0947	0.0601	0.0950	0.0597	0.0946	0.0594
	Hierarchy aspects	0.0997	0.0643	0.0977	0.0647	0.0993	0.0642	0.0979	0.0637	0.0979	0.0633
HetRec ML	Heuristic terms	0.1057	0.0256	0.1105	0.0270	0.1144	0.0277	0.1160	0.0271	0.1174	0.0273
	Classification terms	0.1047	0.0258	0.1125	0.0274	0.1159	0.0280	0.1169	0.0270	0.1180	0.0273
	Heuristic aspects	0.0910	0.0219	0.0991	0.0237	0.1038	0.0246	0.1053	0.0250	0.1054	0.0252
	Hierarchy aspects	0.1062	0.0242	0.1060	0.0262	0.1105	0.0272	0.1143	0.0277	0.1159	0.0260

coherent aspects, but lists of words that may not have explicit relation. This happens since LIHC performs a document hierarchical clustering, instead of words, and then selects the five more representative words of each cluster to compose an aspect. Even so, in some aspects, we can see words that have relation to each other, such as in "[paradis,cinema,camera,screen,past]" with the words "cinema," "camera," and "screen."

Regarding computation time and resources, we argue that the main difference between the approaches lie in the item similarity calculation, since the size of the representations is significantly different. Having the item similarity calculated, which can be done offline periodically, the system will perform within the same time.

Finally, we also argue that our approach can help in minimizing problems regarding new items. In the traditional item *k*-NN, item representations were constructed with the ratings the item has received, thus hurting the prediction of newly added items. Our approach overcomes this drawback by constructing representations that rely on

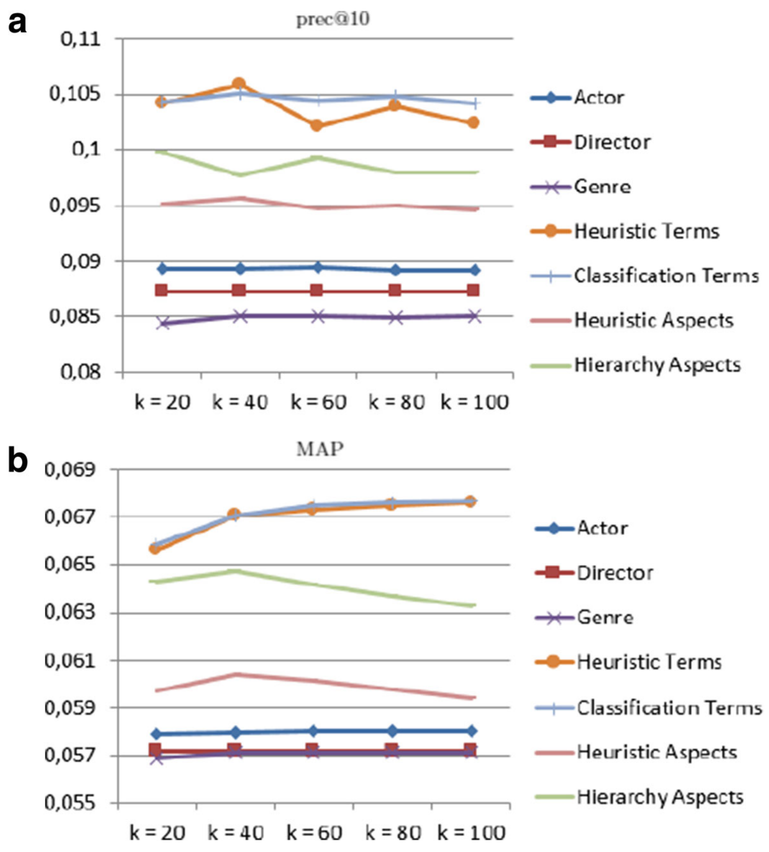


Fig. 6 a, **b** Graphics showing. **a** prec@10 and **b** MAP results for the ML-100k database

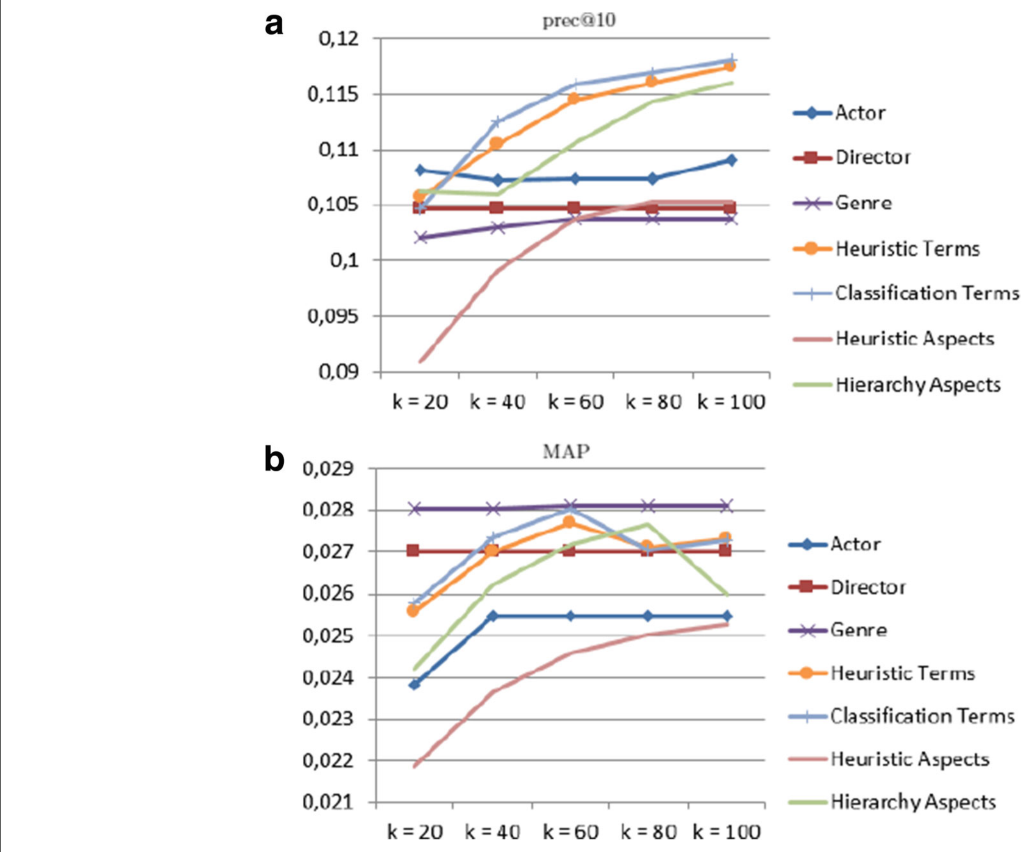


Fig. 7 a, **b** Graphics showing. **a** prec@10 and **b** MAP results for the HetRec ML database

reviews external to the database. In our experiments, we used reviews from IMDb, but reviews from several other websites can be extracted, thus minimizing the chances that an item would remain without any description.

Conclusion and future work

In this paper, we compared four different techniques of text feature extraction for item representation construction and analyzed the impact they produce in a neighborhood-based recommendation algorithm. The results showed that the techniques based on terms provide better results, since they produce a larger set of features, hence detailing better the items. Another point worth noticing is that, for both characteristic granularities, the techniques based on machine learning provided better results. Among all techniques addressed, the technique based on transductive learning provided the best results, being statistically superior in a larger database.

Even though the classification terms technique provided the best results, the other term-based technique also produces interesting results, being significantly easier to implement since it is based on simple heuristics. Those heuristics also require fewer computational resources, being an interesting approach for limited recommender servers. The hierarchy aspects approach also provided interesting results in a larger database setting, providing a much smaller set of features than the term-based approaches, being interesting for systems that performs the item similarity calculation online and requires faster results. Finally, the heuristic aspects approach provided competitive results with the baseline, being a good alternative for systems that have restrictive computational resources and do not have available items' metadata.

As future work, we plan to analyze the proposed set of feature extraction techniques with other recommender algorithms, including those specific for the item recommendation task and optimization of personalized rankings, such as the BPR (Bayesian Personalized Ranking) [31], CliMF (Collaborative Less-is-More Filtering) [32], and TFMAP (Tensor Factorization for MAP Maximization) [33]. Another possibility of extension is to apply the items' representations in different attribute aware recommendation algorithms or to apply the system into different data domains. Finally, we plan to build user profile vectors based on these techniques, by using their own reviews or inferring their sentiment toward the features by analyzing the vectors of the items he/she evaluated as interesting.

Table 7 Top 10 most frequent features extracted from each technique in each dataset

Heuristic terms	Classification terms	Heuristic aspects	Hierarchy aspects
ML 100k			
film	film	cinematography	[discov,homeless,filmmak,maggi,thought]
movie	movi	critics	[paradis,cinema,camera,screen,past]
time	time	horror	[juror,juri,sayl,chang,reason]
character	stori	time	[nichol,listen,happen,experi,mike]
story	charact	scene	[marci,stai,store,leav,slow]
way	scene	audio	[mail,postman,deliv,put,post]
thing	plot	description	[art,form,artist,rylanc,interest]
people	soundtrack	cast	[rylanc,form,atmospher,artist,mind]
scene	plai	footage	[materi,falk,spiritu,chanc,grandfath]
plot	view	script	[carri,spacek,palma,lauri,barn]
HetRec ML			
film	film	cinematography	[comic,final,western,enjoi,fan]
movie	movi	cast	[petti,tank,comic,person,enjoi]
time	time	watch	[saramago,blind,viewer,turn,point]
story	watch	critics	[symbol,imag,view,experi,place]
way	stori	time	[scari,sound,base,run,minut]
people	end	audio	[lucia,reason,kind,artist,art]
character	work	distribution	[declin,spheeri,troop,scout,exploit]
thing	charact	fantasy	[wake,linklat,present,view,understand]
scene	plot	direction	[mckellar,sandra,hour,bit,special]
life	plai	romance	[makoto,chang,travel,van,moment]

We then plan to apply these users' vectors alongside the items' representations in the recommendation process.

Endnotes

[1] https://stanfordnlp.github.io/CoreNLP/.

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

[3] http://movielens.umn.edu.

[4] http://www.imdb.com.

[5] http://grouplens.org/datasets/hetrec-2011.

Abbreviations

BPR: Bayesian personalized ranking; CliMF: Collaborative less-is-more filtering; DF: Document frequency; IF: Item frequency; IMDb: Internet movie database; *k*-NN: *k* nearest neighbors; LIHC: LUPI-based incremental hierarchical clustering; LLGC: Learning with local and global consistency; ML-100k: MovieLens 100k; HetRec ML: HetRec2011 movielens-2k; MAP: Mean average precision NLP: Natural language processing; POS: Part-of-speech; Prec@n: Precision at *n*; RMSE: Root mean square error; TF: Term frequency TF-IDF: Term frequency-inverse document frequency; TFMAP: Tensor factorization for MAP maximization; TLATE: Transductive learning for automatic term extraction; UPGMA: Unweighted pair group method with arithmetic mean; XML: Extensible markup language

Acknowledgements

The authors would like to thank the financial support from São Paulo Research Foundation (FAPESP), CAPES/Brazil, CNPq/Brazil and Araucária Foundation (Paraná/Brazil).

Funding

This work was produced with grants 2013/10756-5, 2013/22547-1 and 2016/20280-6, São Paulo Research Foundation (FAPESP); A075/2013, CAPES/Brazil; and CNPq/Brazil.

Availability of data and materials

The MovieLens 100k dataset supporting the conclusions of this article is available in the GroupLens Web site, https://grouplens.org/datasets/movielens/100k/. The HetRec 2011 ML dataset supporting the conclusions of this article is available in the GroupLens Web site, https://grouplens.org/datasets/hetrec-2011/. Both datasets were incremented with movie reviews extracted directly from the IMDb Web site, http://www.imdb.com/.

Authors' contributions

RMD contributed with the work's proposition and implementation, the experiments' execution and methodology, and the paper's writing. MAD contributed with the work's proposition and the paper's writing and organization. MGM contributed with the paper's proposition, writing, and organization, as well as orientation on the experiments' execution and methodology. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details
[1]Institute of Mathematics and Computer Science, Sao Paulo University, Sao Carlos, SP, Brazil. [2]Department of Informatics, State University of Maringa, Maringa, PR, Brazil. [3]Institute of Mathematics and Computer Science, Sao Paulo University, Sao Carlos, SP, Brazil.

Received: 26 February 2017 Accepted: 19 May 2017
Published online: 05 June 2017

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17(6):734–749
2. Lops P, de Gemmis M, Semeraro G (2011) Content-based recommender systems: state of the art and trends. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds). Recommender Systems Handbook. Springer, Boston. pp 73–105
3. Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds). Recommender Systems Handbook. Springer, Boston. pp 107–144
4. Koren Y, Bell R (2011) Advances in collaborative filtering. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds). Recommender Systems Handbook. Springer, Boston. pp 145–186
5. Yin H, Cui B, Chen L, Hu Z, Zhou X (2015) Dynamic user modeling in social media systems. ACM Trans Inf Syst (TOIS) 33(3):10–11044
6. Ganu G, Kakodkar Y, Marian A (2013) Improving the quality of predictions using textual information in online user reviews. Inf Syst 38(1):1–15
7. Kim HW, Han K, Yi MY, Cho J, Hong J (2012) Moviemine: personalized movie content search by utilizing user comments. IEEE Trans Consum Electron 58(4):1416–1424
8. Qumsiyeh R, Ng YK (2012) Predicting the ratings of multimedia items for making personalized recommendations. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12). ACM, New York. pp 475–484
9. Pero Š, Horváth T (2013) Opinion-driven matrix factorization for rating prediction. In: Carberry S, Weibelzahl S, Micarelli A, Semeraro G (eds). User Modeling, Adaptation, and Personalization: 21th International Conference, UMAP 2013, Rome, Italy, June 10-14, 2013 Proceedings. Springer, Berlin. pp 1–13
10. Aggarwal CC, Zhai CX (2012) Mining text data. Springer, Boston
11. D'Addio RM, Manzato MG (2015) A sentiment-based item description approach for knn collaborative filtering. In: Proceedings of the 30th ACM/SIGAPP Symposium On Applied Computing (SAC '15). ACM, New York. pp 1060–1065
12. D'Addio RM, Manzato MG (2014) A collaborative filtering approach based on user's reviews. In: 2014 Brazilian Conference on Intelligent Systems (BRACIS '14). IEEE, São Carlos. pp 204–209
13. D'Addio R, Conrado M, Resende S, Manzato M (2014) Generating recommendations based on robust term extraction from users' reviews. In: Proceedings of the 20th Brazilian Symposium on Multimedia and the Web (WebMedia '14). ACM, New York. pp 55–58
14. Li H, Li C-h, Zhang S (2009) Learning to recommend product with the content of Web page. In: FSKD '09. Sixth International Conference on Fuzzy Systems and Knowledge Discovery, vol. 7. IEEE, Tianjin. pp 561–565
15. Wang F, Chen L (2012) Recommending inexperienced products via learning from consumer reviews. In: 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT '12). IEEE, Macau. pp 596–603
16. Aciar S, Zhang D, Simoff S, Debenham J (2007) Informed recommender: basing recommendations on consumer product reviews. IEEE Intell Syst 22(3):39–47
17. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Association for Computational Linguistics, Baltimore. pp 55–60
18. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '13). Association for Computational Linguistics, Seattle. pp 1631–1642
19. Porter MF (1997) An algorithm for suffix stripping. In: Sparck Jones K, Willett P (eds). Readings in Information Retrieval. Morgan Kaufmann Publishers Inc., San Francisco. pp 313–316
20. Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York
21. Liu B, Zhang L (2012) A survey of opinion mining and sentiment analysis. In: Aggarwal CC, Zhai C (eds). Mining Text Data. Springer, Boston. pp 415–463
22. Conrado MS, Rossi RG, Pardo TAS, Rezende SO (2013) Applying transductive learning for automatic term extraction: the case of the ecology domain. In: Proceedings of the IEEE Second International Conference on Informatics and Applications (ICIA '13). IEEE, Lodz. pp 264–269
23. Rossi RG, Lopes AA, Rezende SO (2014) A parameter-free label propagation algorithm using bipartite heterogeneous networks for text classification. In: Proceedings of the 29th ACM/SIGAPP Symposium On Applied Computing (SAC '14). ACM, New York. pp 79–84
24. Zhou D, Bousquet O, Lal TN, Weston J, Schölkopf B (2004) Learning with local and global consistency. In: Advances in Neural Information Processing Systems (NIPS '04). MIT Press, Cambridge. pp 321–328
25. Marcacini RM, Rezende SO (2013) Incremental hierarchical text clustering with privileged information. In: Proceedings of the 2013 ACM Symposium on Document Engineering (DocEng '13). ACM, New York. pp 231–232
26. Mitchell TM (1997) Machine Learning. McGraw-Hill, Inc., New York
27. Zhao Y, Karypis G (2002) Evaluation of hierarchical clustering algorithms for document datasets. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management. pp 515–524
28. Domingues MA, Sundermann CV, Barros FMM, Manzato MG, Pimentel MGC, Rezende SO, Oliveira S (2015) Applying multi-view based metadata in personalized ranking for recommender systems. In: Proceedings of the 30th ACM/SIGAPP Symposium On Applied Computing (SAC '15). ACM, New York. pp 1105–1107
29. Koren Y (2010) Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans Knowl Discov Data (TKDD) 4(1):1–1124
30. Bobadilla J, Ortega F, Hernando A, GutiéRrez A (2013) Recommender systems survey. Knowl-Based Syst 46:109–132
31. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: Bayesian personalized ranking from implicit feedback. In: UAI '09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, Arlington. pp 452–461
32. Shi Y, Karatzoglou A, Baltrunas L, Larson M, Oliver N, Hanjalic A (2012) Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the Sixth ACM Conference on Recommender Systems. RecSys '12. ACM, New York. pp 139–146
33. Shi Y, Karatzoglou A, Baltrunas L, Larson M, Hanjalic A, Oliver N (2012) Tfmap: Optimizing map for top-n context-aware recommendation. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '12. ACM, New York. pp 155–164