2015-04

# A sentiment-based item description approach for kNN collaborative filtering

# A Sentiment-Based Item Description Approach for kNN Collaborative Filtering

Rafael M. D'Addio, Marcelo G. Manzato
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
{rdaddio, mmanzato}@icmc.usp.br

## ABSTRACT

In this paper, we propose an approach based on sentiment analysis to describe items in a neighborhood-based collaborative filtering model. We use unstructured users' reviews to produce a vector-based representation that considers the overall sentiment of those reviews towards specific features. We propose and compare two different techniques to obtain and score such features from textual content, namely term-based and aspect-based feature extraction. Finally, our proposal is compared against structured metadata under the same recommendation algorithm, whose results show a significant improvement over the baselines.

## Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*indexing methods, linguistic processing*

## General Terms

Algorithm

## Keywords

Recommender systems, collaborative filtering, item representation, sentiment analysis

## 1. INTRODUCTION

Recommender Systems are a content filtering technology that aims to handle the so called "information overload problem", where users are overwhelmed by the increasing availability of data on the Web [14]. In the collaborative filtering (CF) approach, in particular neighborhood models, the algorithms find and match ratings from clusters of similar users or items to predict unknown ratings. These clusters, in turn, can be computed using correlation measures, which may be based on the ratings themselves, or attributes/metadata related to users or items.

*SAC'15* April 13-17, 2015, Salamanca, Spain.
Copyright 2015 ACM 978-1-4503-3196-8/15/04...$15.00.
http://dx.doi.org/10.1145/2695664.2695747

Regarding the use of metadata to describe the content, there is a growing effort nowadays to consider unstructured data produced by the same or other users to improve the recommendation accuracy [9]. For instance, users' reviews are a rich information source that can support consumers to decide whether it is worth buying or consuming a particular item. Usually, those users manually check such source of information prior the consumption, but automatic techniques could smooth this task by incorporating such analysis into the filtering process, resulting in better recommendations.

However, there is set of challenges that has to be dealt with when using unstructured textual content, in particular users' reviews, for representing items [1]. First, the 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. Opinion mining, or sentiment analysis, is a relatively novel research field that aims to analyze user-authored text, identifying its orientation, which can be negative, positive or neutral/objective. Some of the reviews can also contain sarcastic/ironic sentences, whose identification is also a challenge in the field [8]. Finally, there is a lack of research about how to organize and use the additional data provided by the users in order to enhance items' representations, and consequently, to improve the accuracy of recommendations.

Thus, this paper proposes a collaborative filtering approach which uses users' reviews to better describe items to be recommended. First, the reviews of a variety of users are processed by means of NLP tools in order to extract candidate features and personal sentiment regarding each feature. After that, the algorithm creates an item representation model which contains the average orientation of the users about the selected features. Finally, this representation is used to compute the similarity of items which will be used in a CF approach based on $k$ nearest neighbors. In this work, we used two different approaches to generate features sets: term-based and aspect-based. Using these two approaches, we apply a specific sentiment analysis algorithm to capture the texts' polarities, and we compare them using a dataset of movies recommendation.

This paper is structured as follows: in Section 2 we present the related work about recommender systems based on users' reviews; in Section 3 we describe in details our proposal of describing items based on unstructured data; in Section 4 we present the recommendation algorithm which uses the new items representation model; in Section 5 we present the evaluation and a discussion about the results obtained; and in Section 6 we present the conclusions and future work.

## 2. RELATED WORK

While earlier attempts in content-based filtering make use of structured metadata to describe the content, e.g. genres, list of actors and keywords in a movies recommendation system, more recent works explore the use of unstructured information for the characterization of items.

Reviews authored by Web users are a valuable resource since they provide meaningful and useful semantic information related to the utility of items and/or the preferences of the reviewer [5]. Such information may be related to the item as a whole (e.g. the movie was great) or to specific features (e.g. the actor's performance in that movie was poor). Recent works use these reviews to extract feelings related to inherent characteristics of an item. For instance, Qumsiyeh and Ng [13] proposed a system capable of generating recommendations for different multimedia items, using information extracted from databases available in several trusted sites. Their method is capable of computing the sentiment and degree of each considered aspect of an item: genres, actors and reviews. Kim et al. [6] proposed a personalized search engine for movies, called MovieMine, based on previous reviews of a user and their ratings assigned to other items. In this system, the user types a query that is expanded by the addition of existing keywords in previous reviews.

These previous works use reviews in a content-based scenario, but there are several other works that use textual information in the context of collaborative filtering. For example, Ganu et al. [4] proposed a recommender system based on restaurant reviews that performs a users soft clustering based on topics and sentiments found in their reviews. In this work, it is produced text-based ratings, and their values are compared to the regular star rating system in several scenarios, such as neighborhood and latent factors models. Finally, those two rating systems are compared to the soft clustering recommendation model proposed. Pero and Horváth [12] proposed a framework that uses sentiment analysis to produce text-based ratings that are processed by an matrix factorization algorithm alongside the user's provided ratings. They proposed and compared three different usages for the text-based ratings, namely: pre-filtering, post-filtering and modeling.

Those related works use the user-provided text to derive ratings or to describe the user's preferences, but none of them use reviews to describe the items. As a response, in [2] and [3], users' reviews were used to characterize items to address an item recommendation scenario by producing rankings of items. While in [3] the system used NLP techniques to identify features, in [2] the system relied in a robust term extraction technique which uses semi-supervised machine learning algorithms to select terms that represent the whole set of reviews. Both works scored features as binary sentiment (either positive or negative).

The technique proposed in this paper differs from the aforementioned works since it is a collaborative filtering approach that uses users' reviews and sentiment analysis to solely describe items, focusing in a rating prediction scenario. We compare two different feature extraction techniques, one based in terms and another based on concepts (aspects). In addition, this technique considers different degrees of intensity for each feature's polarity, by scoring its features with floating values.

## 3. PROPOSED WORK

Prior to describing our approach, we adopt the following definitions for both terms and aspects of an item's review. Both are treated as features of items, and both sets are represented as vectors, where each position corresponds to a feature. Aspects consist of nouns ou noun phrases which are very common in the content being analyzed [8]. Terms, on the other hand, correspond to the nouns that are present in the reviews of a certain amount of items; thus, not every term will be present in the whole items set. The aspect-based approach, in turn, considers each aspect as a concept that is bound to be present in every single item. In this sense, each aspect constitutes of a set of words that altogether describes a topic of the items' domain.

Each of these approaches has two different scoring methods, though both are based on sentiment analysis. To compute the scores, we use natural language processing tools to produce vector-based item representations where each position reflects a feature (plot, explosion, etc.), and its score represent the overall sentiment (positive, neutral or negative) towards that feature. In Section 5 we compare those two approaches according to the accuracy of recommendations. In the next subsections, we describe our proposal in details. Subsection 3.1 presents the datasets we used in our research and Subsection 3.2 describes the steps performed to generate the items' descriptions.

### 3.1 Data Acquisition

In this work, we apply our research in a movies recommendation domain, since there are many datasets and additional information available on the Web. In this sense, we combined two different sources of information in order to produce a representative set of items' descriptions: the well-known MovieLens 100k[1] dataset, and the set of movies descriptions provided by the Internet Movie Database[2] (IMDb).

The IMDb is a Web site that contains information about movies. It provides structured metadata, such as genre, length and casting, and also provides unstructured information, such as synopses and user reviews. We collected from this Web site both structured metadata and unstructured content from the movies. In the first case, structured descriptions consist of genres, actors, directors, writers and keywords, which are used as baseline in our experiments. In the second case, we collected users' reviews to be used in our approach. On average, the first 10 reviews for each of the 1,682 movies of the MovieLens dataset were gathered from IMDb, resulting in a total of 15,863 text documents. It is worth mentioning that not all movies had 10 reviews. Some movies did not have reviews on the IMDb, so other trusted sources, such as the Rotten Tomatoes [3], were used to complete those missing reviews.

### 3.2 Data Modeling and Processing

As previously exposed, item' representations are generated by modeling the data available into vectors where each position represents a particular feature. To accomplish this,

[1]http://grouplens.org/datasets/movielens/
[2]http://www.imdb.com/
[3]http://www.rottentomatoes.com/

we rely on NPL tools that process the reviews and transform the unstructured content into a more structured representation.

Thus, to process the texts we use the free, open source Stanford's natural language processing tool, called Stanford CoreNLP[4] [11]. This framework contains the most common NLP routines implemented as annotators. In our work, we used the following annotators: tokenizer, sentence splitter, lemmatizer, part-of-speech (POS) tagger [17], parser [15] and the sentiment analysis tool [16]. For each text file, it outputs a graph-form XML file with all set of relevant annotations previously defined.

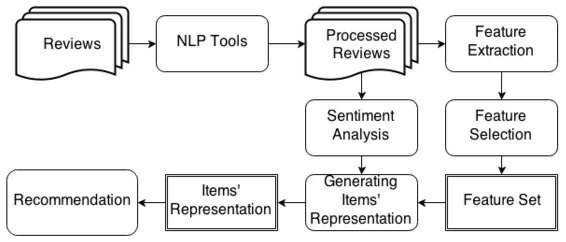

**Figure 1: The process for generating item representations.**

Figure 1 illustrates the overall steps to generate the items' representations based on users' reviews. The reviews are processed with the Stanford CoreNLP tool, which outputs a set of XML files. In each XML document, the content is structured in a set of sentences, which, in turn, have the related sentiment and parser. In addition, the sentences are split into tokens and each of them has its lemma and its POS tag. These pieces of information are then processed by the Feature Extraction and the Feature Selection modules, which are responsible to extract candidate terms and to select only the relevant terms that will constitute the Feature Set.

In order to compare the results, we implemented two different feature extraction and selection approaches: the first one is based on terms, and is described in Subsection 3.2.1; and the second is based on aspects, and is presented in Subsection 3.2.2. Regardless of the feature extraction and selection approach, we construct the items' representations (Section 3.2.3) by considering the feature set and the sentiment of the sentences provided by the Stanford sentiment analysis tool, which is executed alongside as an annotator of the Stanford CoreNLP. As final step, we provide the representations to the CF module, which will compute the recommendations.

### 3.2.1 Term-Based Feature Extraction

The first approach selects terms from the reviews and use them directly as the features of the item. There are many heuristics to select the most relevant terms from documents [10]; in this paper, since we're dealing particularly with one data domain, i.e. reviews about movies, and the features are often nouns such as "effects", "plot" or "direction", we rely only on the POS tag to select our features. In this sense, in the feature extraction module, we extract as candidate features only the terms with their part-of-speech tag corresponding to singular and plural nouns.

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

One particularity of the Stanford CoreNLP POS tagging is that misspelled words tend to be tagged as nouns. Consequently, a great number of candidate features are just noise that would affect the accuracy of the recommender system. In addition, the more (irrelevant) features are considered to describe items, the more sparse will be the item vs. feature matrix, resulting in higher computational cost. To tackle these problems, we perform a feature selection based on the *item frequency* of a feature [3]. By accomplishing this task, we are able to remove terms that are not very frequent, and consequently, less important, in relation to the whole set of items.

Let $F$ be the vocabulary and $I$ the set of items; the item frequency $IF_f$ of a feature $f$ is given by Equation 1:

$$IF_f = \sum_{i}^{|I|} k_{if}, \tag{1}$$

where $k_{if}$ is equal to 1 if an item $i$ contains that feature or 0 otherwise.

The $IF_f$ is then compared with a threshold $t$ to decide if the feature is removed or maintained in the vocabulary: if the value of the $TF_f$ is higher then $t$, the feature should be maintained in the vocabulary. We tested with many different thresholds and the best result for the considered dataset is $t = 30$.

The resulting set of features is then processed by the item representation generation module, which is detailed in Section 3.2.3.

### 3.2.2 Aspect-Based Feature Extraction

In addition to the previously described term-based feature extraction approach, we generated features based on different aspects which are frequently mentioned in the reviews. Concretely, instead of extracting features based on individual terms, we considered a set of terms to describe one single feature. For instance, the aspect **direction** would constitute of words such as **director**, **directing**, **direct**, among others.

To accomplish this task, we stemmed the words using the Porter algorithm [10], since one stem relates to a set of words with different POS tag that are related to a same topic, and kept a record for which word each stem points to. Then, we converted all letters to lowercase; and removed stop words, special characters, punctuation, numbers, accents, and words composed of only one character. In this way, we presume that most of the data noise is removed.

In the next step, we clustered words which had the same or a closely related stem into the same category. We also clustered stems of synonym words, e.g., **plot** and **storyline**. This step was semi-supervised in order to reduce the occurrence of errors in the process.

We also executed the feature selection process, similarly with a threshold of $t = 30$, in order to remove the term sets that had little occurrence in the texts, and thus, would interfere negatively the results. At this point, the main difference is that while the previous feature selection considered only a term as a feature, in this approach it considered aspects with more than one term as features.

The resulting set of aspects is then processed by the item representation generation module in a similar way as the term-based approach, which is detailed in the next subsection.

### *3.2.3 Setting Scores for Features*

To generate the items' representation, we used a vector-based approach to describe the items, where each position represents a feature from the feature set generated in the previous step. In this sense, we build an item vs. feature matrix with scores, where each score reflects the overall sentiment of the item's reviews towards a determined feature.

To compute the sentiments for each topic in each review, we use the Stanford sentiment analysis tool [16], an annotator of the Stanford CoreNLP. This sentiment analysis tool has a novel approach since it uses a deep learning model that is capable of using the sentence structure to provide sentiment analysis in a sentence level. In this sense, this tool computes the sentiment based on the meaning of each word considering its context in the sentence. Since most movie features are nouns, and nouns are generally neutral sentiment words, it is a good idea to use a sentiment analysis tool that relies on sentence-level scoring. Instead of assigning a polarity by checking the isolated word's sentiment, which would lead us to several features with a neutral value, the score is assigned based on the sentiment of the sentence that contains that word. Another justification is that this model can deal with negation sentences, which often presents a set of negated positive words.

As soon as we have the sentiment of the reviews' sentences, we can compute the score of the features by considering the sentences related to those features. For each feature in each item, we cluster the sentences related to them and analyze their sentiment. The Stanford CoreNLP sentiment analysis tool classifies sentences in five sentiment levels: Very Negative, Negative, Neutral, Positive and Very Positive. We converted this classification into a $[1;5]$ rating system and assigned as a feature score the average rating of the related sentences. A zero value indicates that an item doesn't have that feature. A 3 value indicates that the item's reviews have an average neutral sentiment towards the feature, while a value higher than 3 indicates that there's a positive sentiment, and a value lower than 3 indicate a negative sentiment.

In our experiments, we also generated a binary matrix where each feature is divided into two columns, a negative part and a positive part. We set the negative part as 1 if the overall sentiment is negative (lower than 3), and set the positive part as 1 if the overall sentiment is higher than 3. Both approaches are compared in Section 5.

## 4. THE RECOMMENDER ALGORITHM

The last step of our proposal is to apply the novel reviews-based items' representation into a recommender system algorithm. A state-of-art collaborative filtering [7] was used, which computes similarities among items according to the traditional item-based neighborhood model ($k$-nearest neighbors). It is known that this algorithm has higher accuracy because it also takes into account the user and item bias [7]. We extended this model so that the items' similarities are now dictated by the items' description vectors, instead of originally considering the items' ratings vector. In this section, we briefly describe the model; a more detailed description can be found in [7].

A common approach of CF 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 $\mu$ is the global average rating, $b_i$ 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 an user $u$ to an item $i$. In our experiments, we iterate 10 times these equations following the order they appear, and set the constants $\lambda_1$ e $\lambda_2$ as, respectively, 10 and 15.

The goal of the recommender 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 items. It can be based on several correlation or distance metrics, such as the Pearson correlation coefficient, $p_{ij}$, which measures the tendency of users to rate items $i$ and $j$ similarly [7]. Another similarity measure is the cosine correlation, used mainly in the information retrieval area and in content-based algorithms [9]. We performed tests in both correlation metrics, and found out that the Pearson correlation coefficient presented better results. The final similarity measure is a shrunk correlation coefficient, $s_{ij}$:

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_3} p_{ij}, \tag{5}$$

where $n_{ij}$ is the number of features that describe both items $i$ and $j$, and $\lambda_3$ is a regularization constant, set as 100 [7].

Given that, we identify the $k$ items rated by $u$ that are most similar to $i$, the $k$-nearest neighbors. 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 estimate:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j\in S^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j\in S^k(i;u)} s_{ij}}. \tag{6}$$

## 5. EXPERIMENTAL EVALUATION

In order to evaluate our proposal, we apply the items' representation based on users' reviews into the neighborhood-based recommendation algorithm described in the previous section. We aim to evaluate how the model could improve the recommendation results in a rating prediction scenario, i.e., how well the algorithm can predict ratings for unseen items, given a set of already rated items. For that, we evaluate our algorithm using the root mean square error (RMSE) measure [14], where it evaluates the discrepancy of a predicted rating to its real value. We performed a 10-fold cross validation where for each fold the data is divided into 90% for training and 10% for testing.

As baselines, we compare our results with structured metadata obtained from IMDb. The baseline are binary matrices that represent whether the item has or not determined metadata, such as: genres, writers, directors, actors and keywords. It is worth mentioning that the keywords are not extracted from the reviews; they are the keywords listed in the IMDb dataset.

As described in Section 3, our item matrices consist of two term-based and two aspect-based set of features: one is a binary sentiment matrix and the other is a float-valued matrix with scores between the [1; 5] interval, which we will call as rating matrix.

Table 1 presents the overall results (rounded to the fifth decimal case) obtained in the cross-validation evaluation for $k = \{20, 40, 60, 80, 100\}$. Figure 2 compares the term-based and the aspect-based matrices in both proposed scoring types: the binary sentiment matrix and the [1;5] rating matrix.

As it can be seen, the term-based approach obtained better accuracy than the aspect-based in both matrices. This implies that a larger set of features is bound to describe the data better than a significantly smaller set. While the term-based approach contains 3,085 features, the aspect-based approach contains only 41 features.

In the term-based approach, it can be seen that as the number of neighbors increases, the accuracy of the recommendations decreases. One possible reason is that as the items are described in more details, there is a chance that, as the number of neighbors increases, there will be items among the neighbors that are not very close to the item whose rating is being predicted.

The opposite effect can be seen in the aspect-based approach, where as the number of neighbors increases, the results are better. We argue that it happens because these approaches have very little features to differentiate the items, thus needing more neighbors to correctly predict ratings.

Another issue that can be observed in Figure 2 is that while the rating-based scoring produces better results for the term-based approach, the opposite occurs with the aspect-based approach. Similarly, the binary scoring produces worse results for the term-based approach and better results for the aspect-based approach. We argue that since the binary scoring duplicates the number of features (each feature is now treated as having positive and negative binary scores), the aspect-based approach has bigger vectors to describe the items, thus producing better results. The term-based approach, in turn, already have a large set of features to describe the items and, by duplicating it, produces very sparse vectors resulting in a worsening of the results.

Considering only the [1;5] rating matrices, in Figure 3 we show the times of correlation and recommendation phases of both term-based and aspect-based approaches. As it can be seen, the times of recommendation phase were almost the same for both approaches, according to the number of neighbors. On the other hand, the major difference lies in the correlation times, where using the term-based matrix it performed in around 22000 milliseconds, and using the aspect-based matrix it performed in around 300 milliseconds. Considering the RMSE results showed in the Figure 2, we argue that although the correlation times among term and aspect-based approaches were significantly different, the term-based approach is still preferred over the aspect-based, as recommendations are computed in similar times, and the correlation, in turn, could be computed offline.

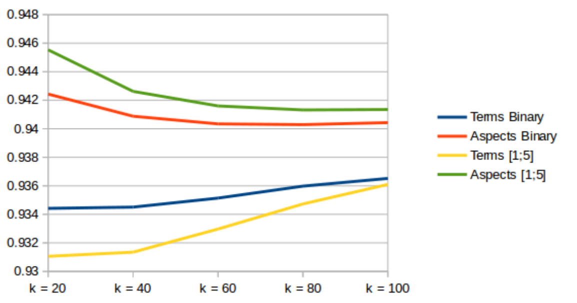

**Figure 2: Graphic comparing the RMSE results for the binary and [1;5] rating matrices of both feature sets produced.**

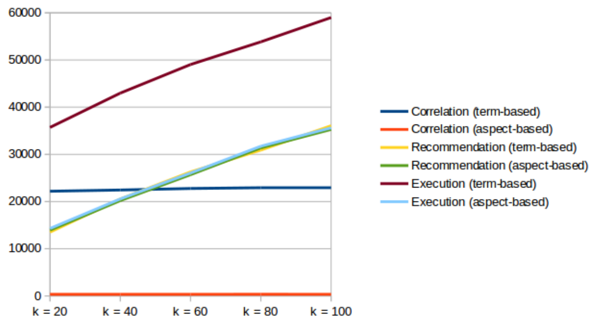

**Figure 3: A comparison between aspect and term-based approaches concerning the execution time, in milliseconds.**

In Figure 4, we compare the term-based approach (binary and rating matrices) with the baselines. As it can be seen, our approach obtained the best results when compared to structured metadata, where the [1;5] rating matrix had the best results than all other matrices. We performed a Student T-test to evaluate if our approach yields statistically significant improvement and found that indeed it has better results when compared to all metadata types ($p$-value $< 0.005$).

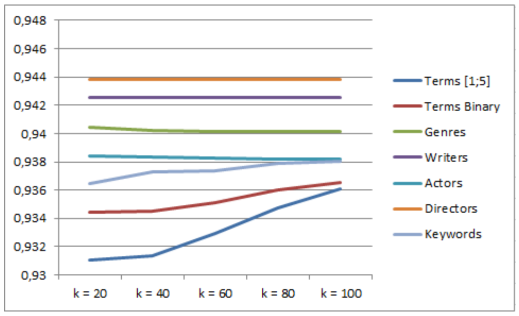

**Figure 4: Graphic comparing the term-based approach with the baseline metadata matrices.**

Still concerning Figure 4, one may note that for most of the metadata matrices, almost no RMSE variation could be

**Table 1: The RMSE results for the metadata baseline and the matrices produced in the proposed approaches.**

| | | k = 20 | k = 40 | k = 60 | k = 80 | k = 100 |
|---|---|---|---|---|---|---|
| Metadata | Genres | 0,94041 | 0,94018 | 0,94015 | 0,94013 | 0,94012 |
| | Writers | 0,94253 | 0,94253 | 0,94253 | 0,94252 | 0,94252 |
| | Directors | 0,94384 | 0,94385 | 0,94384 | 0,94384 | 0,94384 |
| | Actors | 0,93843 | 0,93833 | 0,93825 | 0,93821 | 0,93816 |
| | Keywords | 0,9365 | 0,93726 | 0,93734 | 0,93787 | 0,93805 |
| Binary Scores | Aspect Approach | 0,94243 | 0,94088 | 0,94034 | 0,94029 | 0,94043 |
| | Term Approach | 0,93442 | 0,93451 | 0,93514 | 0,93598 | 0,93652 |
| Rating Scores | Aspect Approach | 0,94553 | 0,94262 | 0,9416 | 0,94132 | 0,94135 |
| | Term Approach | 0,93106 | 0,93135 | 0,93297 | 0,93473 | 0,9361 |

observed at different $k$ numbers. This behavior is due to the sparseness of the matrices. For instance, the actor baseline has around 50,000 features, and only a small portion of that constitutes each of the movies' cast. As for smaller sets of features, such as genres and the aspect-based approach, we observe that as the number of neighbors increases, the results are better. As discussed above, it happens because these approaches have a very small set of features to characterize the items, thus needing a bigger number of neighbors to correctly predict ratings.

## 6. CONCLUSIONS

In this work we proposed an items' representation approach based on sentiment analysis of users' reviews. We explored two different feature extraction techniques and compared both in a rating prediction scenario. One important finding is that the term-based approach, which produces a significantly larger feature set, yields better results than an aspect-based approach.

In future work we plan on evaluate the same approaches in much larger datasets and reviews. We also plan to apply these approaches to other item attribute aware recommendation algorithms and measure the differences those approaches can cause in the recommendation process.

## 7. ACKNOWLEDGMENTS

The authors would like to thank the financial support from FAPESP (process numbers 2013/10756-5, and 2013/22547-1).

## 8. REFERENCES

[1] C. Aggarwal and C. Zhai. A survey of text clustering algorithms. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 77–128. Springer US, 2012.

[2] R. D'Addio, M. Conrado, S. Resende, and M. Manzato. Generating recommendations based on robust term extraction from users' reviews. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, WebMedia '14, pages 55–58, New York, NY, USA, 2014. ACM.

[3] R. D'Addio and M. Manzato. A collaborative filtering approach based on user s reviews. In *Proceedings of the Brazilian Conference on Intelligent Systems (BRACIS)*, page (to appear), 2014.

[4] G. Ganu, Y. Kakodkar, and A. Marian. Improving the quality of predictions using textual information in online user reviews. *Inf. Syst.*, 38(1):1–15, Mar. 2013.

[5] M. Hu and B. Liu. Mining and summarizing customer reviews. In *PROC of the 10th ACM INT CNF on Knowledge Discovery and Data Mining (SIGKDD)*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[6] H. Kim, K. Han, M. Yi, J. Cho, and J. Hong. Moviemine: personalized movie content search by utilizing user comments. *IEEE Transactions on Consumer Electronics*, 58(4):1416–1424, 2012.

[7] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1:1–1:24, Jan. 2010.

[8] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In C. C. Aggarwal and C. Zhai, editors, *Mining Text Data*, pages 415–463. Springer US, 2012.

[9] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[10] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[12] S. Pero and T. Horváth. Opinion-driven matrix factorization for rating prediction. In S. Carberry, S. Weibelzahl, A. Micarelli, and G. Semeraro, editors, *User Modeling, Adaptation, and Personalization*, volume 7899 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2013.

[13] R. Qumsiyeh and Y.-K. Ng. Predicting the ratings of multimedia items for making personalized recommendations. In *PROC of the 35th Annual INT ACM CNF on Research and Development in Information Retrieval (SIGIR)*, pages 475–484, New York, NY, USA, 2012.

[14] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.

[15] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *PROC of the ACL CNF*, 2013.

[16] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *PROC of the CNF on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics, October 2013.

[17] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.