ScriBa: a script-based recommender system for movies

Giulia Corsi
DISI-University of Trento
Via Sommarive, 9
38123 Trento, Italy
giulia.corsi@studenti.unitn.it

Gianvito Taneburgo
DISI-University of Trento
Via Sommarive, 9
38123 Trento, Italy
gianvito.taneburgo@studenti.unitn.it

ABSTRACT

In this paper we present ScriBa, a content-based recommender system for movies and television series that makes use of plots, synopses, and actors scripts to predict recommendations for users. The data set used to test the recommender system has been taken out from three different sources: Netflix training data set, Internet Movie Database (IMDb) and OpenSubtitles. Results show that the idea of using in-depth text analysis can be a useful similarity criterion. The recommender system obtained with this strategy has succeeded in predicting unknown ratings with acceptable values of root-mean-square error (RMSE).
However, for more effective results on predictions and to avoid the cold start problem, this approach needs to be enhanced also by other strategies, *e.g.* user-user collaborative filtering or content-based boosted with other features such as those regarding actors, genre, director.
The idea underlying a possible hybrid recommender system is presented, although its implementation is left as future work.

CCS Concepts

•**Information systems** → **Recommender systems;** *Content analysis and feature selection; Test collections; Retrieval effectiveness;*

Keywords

Data mining; movies recommendation; content-based filtering; text analysis; hybrid recommender

1. INTRODUCTION

The importance of recommender systems for on-line service providers, as well as their growing presence in everyday life, has lead to new in-depth studies on this subject. A recommender system is an application that involves predicting user responses to options [7] in order to guide them in a personalized way to interesting objects, in a large space of possible options [9]. This prediction can be effectuated principally through two different approaches: content-based and collaborative filtering. What mainly distinguishes one from the other are the subjects on which the similarities are computed.
A content-based recommender is focused on items, compared through the similarity of their properties. A user-user collaborative filtering recommender system is focused on users, whose relationships are determined by the similarity of ratings given by both the users in comparison.
A particular case is the item-item collaborative filtering, also called item-based, which examines the similarity between items, calculating it by using user's ratings on those items. This approach was published in 2001 and it is actually used by Amazon.

Contribution. Commonly, content-based recommender systems for movies and tv series are based on the similarity between different feature sets like those regarding actor, director, genre and keywords [11]. In this paper we want to observe the impact of in-depth text analysis, applied on plots, synopses, and actors scripts, on recommendations.

Related work. In the last ten years, the appreciation gained from on-line services providers opened the gates for investments into recommender systems research area, both in the industrial and in the academic world. A significant boost was given from Netflix, starting from 2006. The American colossus of on-demand streaming media held a data mining competition offering a prize of $1,000,000 to the first team able to beat their own recommender algorithm, the CineMatch. The prize was won by the team "BellKor's Pragmatic Chaos" [6] in 2009.
Despite the success of content-based and collaborative filtering, there are still several limitations. Associated with the content-based approach there are problems such as: difficulty in finding appropriate features to consider, cold-start problem for new users, overspecialization and over attachment to previous knowledge that brings to a lock in same topics [1]. With the collaborative filtering approach some problems are resolved: this method works for any type of item, without the need to perform a feature selection, and is able to include different areas of interest in recommendations. However, there is still a problem of cold-start increased by a bias for popular items and an almost null consideration for new entries. For such reasons, even if collaborative filtering techniques remain the most popular and widely used in practice [10], researchers are doing continue efforts to combine both content-based and collaborative filtering approaches to improve predictions. The most important methods for hybrid filtering that can be identified are the approaches commonly noted as weighted, switching, mixed, feature combination, and cascade [4].
Besides what has been said so far, both for collaborative filtering and content based approaches, the big sparse matrices involved bring space and scalability problems [2]. This issue is stressed in user-user collaborative filtering, in which on-line computation depends on the number of customers, making the algorithm impractical on large data sets [8]. Possible solutions, such as dimensionality reduction, sampling,

or partitioning would reduce recommendation quality and are often considered as unacceptable by the majority.

Structure of the paper. The paper is structured as follows. Section 2 exposes the data collection procedure and the pre-processing pipeline followed. Section 3 introduces the problem statement and the text-based solution. Section 4 discusses the experiments made to test the goodness of the script-based approach and presents the quality results obtained. Section 5 describes how ratings are predicted for the recommender system. Section 6 exposes recommendations result and their correspondent RMSE value. Section 7 presents future work proposals, including the hybrid recommender system. Section 8 concludes the paper.

2. DATA COLLECTION AND PRE-PROCESSING PIPELINE

To collect the required data we made use of three sources: the Netflix training data set, Internet Movie Database (IMDb) and OpenSubtitles. The training data set provided by Netflix during the Netflix Prize competition is composed of 100,480,507 ratings that 480,189 users have given to 17,770 movies [3]. IMDb is an on-line database containing information related to movies and television programs, including plots and synopses of films and series, which have been used in the recommender. OpenSubtitles is an on-line community-driven database in which there are multiple subtitles for movies and series with different formats and languages, sometimes divided in more than one part (CDs). We wrote a Python script to collect the data from these sources and store them in a format suitable for later analyses.

2.1 From Netflix to IMDb

As a first step, each movie on the Netflix data set has been associated with its IMDb identifier. In the Netflix data set movies are represented as a triplet of the form

<MovieID,YearOfRelease,Title>

where:

- the MovieID does not correspond to the IMDb identifier of the corresponding movie nor to the actual Netflix one, but is an random identifier generated for the data set;
- the YearOfRelease may correspond to the DVD release or to the theatrical one;
- the Title is Netflix movie title.

The script has retrieved the IMDb identifier for each movie on Netflix data set searching it by Title and Year on IMDb database. Unfortunately, since this information is ambiguous and some movies are very old, it has been impossible to match the entire data set. However, a portion of 9,939 movies, for which a correct unique match has been found, can be considered sufficient to test the recommender system. Some multiple match for important movies or series have been manually fixed.

2.2 Subtitles download

The previous result allows the second data collection step, that refers to the subtitles search on OpenSubtitles, using its API. Through the IMDb identifier of each matched movie, the Python script has searched for the English subtitle in .srt format divided into the minimum number of CDs. Subtitles for 6,117 movies have been found and downloaded; this number represents the total amount of elements that has been considered during the evaluation of the script-based recommender system. Each subtitle has been cleaned from timecodes, sequential numbers, and possible HTML tags, in order to keep only the text. Subtitles divided into multiple CDs has been unified together.

2.3 Plots and synopses

The last data needed for the content-based recommender system are plots and synopses. They have been requested using the IMDb identifier contained into the list of movies for which a unique correct match on IMDb and an available subtitle have been found. In the case of multiple plots, all of them have been added to the file containing the full text used for the analysis. Plots and/or synopses have been found for 6,104 movies.

2.4 Netflix ratings

The data collection pipeline ends with the extraction of ratings from the Netflix training data set. The respective directory contains one file for each movie with all the ratings given by users to it. In each rating file the first line contains the sequential MovieID. In the subsequent lines, ratings are represented as a triplet of the form

<CustomerID,Rating,Date>

where:

- CustomerIDs range from 1 to 2,649,429 (with gaps since users are 480,189);
- ratings are on a five star scale from 1 to 5 [3];
- dates have the format YYYY-MM-DD.

3. SCRIPT-BASED RECOMMENDER

In this section the content-based filtering is described, in particular, for movies recommendation purpose. The core idea of the content-based approach is to suggest users items similar to those they have previously rated. For each item i a profile is built with a set of distinguishing features previously identified. Those features are summarized in vectors $\vec{v}_i$. A very important task is to understand which particular features are related to the fact that item is appreciated or not. This study is concentrated on text features. The scope of this paper is trying to discover if an in-depth text analysis can be effective in finding similarities in a content-based filtering process.

3.1 Text analysis

In the case of text analysis, heuristics from text mining can be used, namely, the TF-IDF score [5], in order to identify important features. In this way, it is possible to compare documents using techniques such as Jaccard or cosine similarity distance measure. In this paper some basic information about the text analysis are given, without going too deep into this topic.

3.2 Text pre-processing

Before calculating the TF-IDF score of each word in documents, they need to be pre-processed. Each document

is, in this case, for each movie, the union of plot(s) and synopsis(es) from IMDb and the full subtitles text from OpenSubtitles. To perform the text analysis we have used Scikit-Learn, a Python library containing efficient tools and algorithms for data mining and data analysis. The preprocessing step applies some modifications on text such as tokenization, stop-words removing, case folding, and lemmatization. Using the *TfidfVectorizer* available in Scikit-Learn, with its default settings and specifying to use the English dictionary for stop-words identification, it has been possible for the script to obtain a matrix of TF-IDF features.

3.3 TF-IDF score

The TF-IDF confers a score to each word in a document (item i of documents set D). The item profile is made of the set of words with the highest TF-IDF score. The TF-IDF is calculated as:

$$TF\_IDF(w,i) = TF(w,i) \cdot IDF(w)[5]$$

where $TF(w,i)$ calculates the frequency of word w in document i and $IDF(w)$ aims to give a lower weight to those words that appear in a lot of documents. IDF determines the inverse document frequency of each word w, considering the number of documents in which the word appears, n_w, and the total number N of documents, calculating:

$$IDF(w) = log\frac{N}{n_w}.$$

Repeating this step for all text files in D, the result is a matrix in which each document i is represented by a real-valued vector of TF-IDF weights.

3.4 Cosine similarity

In this paper the cosine distance has been used to measure the similarity between documents. This technique takes as input two item profile vectors, $\vec{v}$ and $\vec{u}$, containing the TF-IDF score of each word, and computes:

$$\cos(\theta) = \frac{\vec{v}\cdot\vec{u}}{\|\vec{v}\|\|\vec{u}\|} = \frac{\sum_{i=1}^{n} v_i u_i}{\sqrt{\sum_{i=1}^{n} v_i^2}\sqrt{\sum_{i=1}^{n} u_i^2}}\ [7],$$

where v_i and u_i are components of vectors $\vec{u}$ and $\vec{v}$ respectively representing their attributes.
This operation measures the cosine of the angle between two given vectors.

4. TEXT BASED APPROACH: EXPERIMENTS AND RESULTS

Before completing and testing the recommender system, we have done experiments to evaluate the goodness of the text based approach in finding similar movies. For each movie at least the first five result with higher cosine similarity values have been extracted. Experiments have been done on a large set of movies; however, just some significant examples are presented.

4.1 Series

First of all the system has been tested on series, supposing to obtain very high similarity values between different episodes.

Table 1: Results for: The Lord of the Rings (LOTR): The Return of the King

Title	IMDB id	Similarity
The Lord of the Rings	77869	0.7738
LOTR: The Fellowship of the Ring	1207	0.7715
LOTR: The Two Towers	167261	0.7227
The Return of the King	79802	0.6283
The Day After Tomorrow	319262	0.1170

Table 1 exposes results for the trilogy "The Lord of the Ring". In that case the text similarity obtained has been very high (>0.6) with the first four suggested movies, that represent the other episodes of the trilogy in different versions (extended and cartoon). The similarity falls down to ≈ 0.12 with the fifth movie suggested, "The Day After Tomorrow", which is very different but, anyway, contains similar adventure themes such as surviving and destruction of cities. This last result could seem out of topic with respect to "The Lord of the Rings", but considering the reduced set of available movies it is an appropriate result.
More than other ten series have been analyzed, and all have revealed a similar behavior with respect to the text based system. Indicatively, series and different versions of the same movie hold the higher similarity values when compared to external movies.

Table 2: Results for: Generations (ST VII)

Title	IMDB id	Similarity
Nemesis (ST X)	253754	0.3203
Insurrection (ST IX)	120844	0.2895
The Undiscovered Country (ST VI)	102975	0.2624
First Contact (ST VIII)	117731	0.2283
The Final Frontier (ST V)	98382	0.2098
The Search for Spock (ST III)	88170	0.1701
The Wrath of Khan (ST II)	84726	0.1608
The Motion Picture (ST I)	79945	0.1528
Babylon 5: A Call to Arms	146455	0.1482
The Voyage Home (ST IV)	92007	0.1423

Examining the case of Star Trek it has been observed that, when long series are involved, for episodes very far from one another the similarity can decrease to very low values. Specifically, for an episode e, the similarity of other episodes holds acceptable values for those that follow e, whereas, for previous episodes, the more we go back in time with respect to e, the more a negative trend is registered. Precisely, in Table 2, representing episode VII of Star Trek, results are:

- values >0.2 for episodes from V to X
- values <0.2 for episodes from I to IV

A similar behavior has been observed in other series for which more than five episodes are on the final list of available movies. We can suppose this happens because characters and situations of successive episodes are maintained,

whereas those about previous movies are discarded and forgotten. However, too few examples of long series are available to make strong conclusions about this correlation. In the first ten results of Table 2 also an external movie has been found, "Babylon 5", that maintains the topic of space wars.
Arises spontaneously to ask why the "Star Trek" series has, in general, lower values of similarity in comparison with "The Lord of the Rings" trilogy. This can be explained by the fact that "The Lord of the Rings" uses a very specific dictionary, that is sustained by a high IDF value in the TF-IDF score, whereas in "Star Trek" a lot of words (*e.g.* "crew", "navy", "ship") are shared not only with a lot of other space-battle movies, but also with classic navy movies set on Earth. In fact, after the first 10 suggestions of Table 2, a big set of marine-battle movies has been obtained.
As this fact suggests, even if there are lot of words from the space naval dictionary that are supposing to characterize "Star Trek" series, their recurring presence in a lot of movies causes an important lowering on their IDF value.

4.2 Single movies

Table 3: Results for: Gladiator (2000)

Title	IMDB id	Similarity
The fall of the Roman Empire	58085	0.3722
Caesar	284741	0.3567
Spartacus	54331	0.2635
Sebastiane	75177	0.2261
I, Claudius	74006	0.2236

Switching the subject to single movies, Table 3 exposes results for "Gladiator" (2000) a movie about the drama of a Roman loyal general, betrayed by the Emperor's son, that fight for justice and revenge.
The first three results, with a similarity higher than 0.25, are movies about heroes, honor and fights, all set in Rome. The last two results, with a similarity ≈ 0.22 have some differences (*e.g.* bibliography genre) but, in any case, they contain themes about the Roman Empire, war, love, honor and drama.

Table 4: Results for: The Little Mermaid

Title	IMDB id	Similarity
The Little Mermaid II	240684	0.5341
Outrageous Fortune	93690	0.2140
Sweet Charity	65054	0.2074
200 Cigarettes	137338	0.2053

Changing topic, in Table 4 results are presented also for the cartoon "The Little Mermaid", the Disney production based on the Danish fairy tale with the same name, written by Hans Christian Andersen. The first proposal obtained has been the second episode of the same cartoon, as expected, with a high similarity value.
However, what attracts more are the other results generated with a similarity higher than 0.2, all containing the love theme. In particular, the first two movies are about passing obstacles and dangers to pursue the loved one. Therefore, it seems the approach has worked well in finding similar themes.

Table 5: Results for: Shark Attack 2

Title	IMDB id	similarity
Jaws	73195	0.5172
Shark Attack 3	313597	0.4505
Matchstick Men	325805	0.3558
Are We There Yet?	368578	0.3519
High Sierra	33717	0.3311

Unfortunately, results have not been so good in other cases. The approach seems to have a counter-productive sensibility for proper nouns. As example, in Table 5 results for "Shark Attack 2" are presented. In this movie there are two important characters named Nick and Roy. The first two suggested movies with highest similarity values are the successive episode, "Shark Attack 3", and "Jaws", another movie on a big shark that terrorize a population. In contrast, other three suggestions given, all with similarity values higher than 0.3, are out of the principal topics (sharks, wild animals, ocean).
The problem is related to the correspondence given by proper nouns of protagonists (alternatively Nick and Roy) that are often repeated in scripts, increasing the similarity calculated between movies.
However, those movies are not completely a wrong choice, since they maintain some themes such as adventure and danger.

Table 6: Results for: The Mummy Returns

Title	IMDB id	Similarity
Panic	194218	0.2522
The Big Chill	85244	0.2276
Shallow Grave	111149	0.2165
The Order	304711	0.2019
House on the Edge of the Park	80503	0.1962

As a second example of this problem, results obtained for "The Mummy Returns" are presented. This is an adventure movie in which a family fights against a resurrected mummy and its soldiers to impede the return of the Scorpion King and the army of Anubis.
Results are not very bad in terms of common themes (*e.g.* kidnapping, murder, family in danger, reunion), but they are surely influenced by the name Alex, recurring as a protagonist in all cases.

4.3 Consideration on text based approach

From this experiments the following characteristics of in-depth text analysis system have been observed:

- The approach is very good in finding episodes of series or alternative versions of the same movie, giving them higher values of similarity with respect to other movies. In particular, for an episode e of a series, the system obtains higher values of similarity for episodes closer to e, supporting in particular successive ones. Special series, with a very specific dictionary (*e.g.* "The Lord of

the Rings", "Harry Potter"), are promoted by very high similarity values because of their IDF score, whereas series set in more common locations have, respectively, lower values.

- The approach has reached its purpose in finding similar items: themes and contents are maintained among recommended movies, with a more profound meaning than what could be suggested using just a simple keywords similarity approach. However, the system is not perfectly precise and could be helped by additional information such as the introduction of features like genre, actors, and directors, weighted during the similarity score calculation in an appropriate way.

- There is a counter-productive influence of proper nouns on the similarity score given by their repetition. Even if this feature is not able to alter completely the results, its presence confuses the similarity system. We suggest as a possible solution to include a list of proper nouns together with stop-words, in order to filter them out during the text pre-processing phase.

We decided to consider as good recommendation only movies with a similarity value higher than 0.2.

5. MAKING PREDICTIONS

Since the goodness of the text-based approach has been considered acceptable and, in some cases, a very good element to find similar items, it is possible to use it for recommendations. Known preference values are contained in a sparse utility matrix. The goal of the recommender system is to predict the blanks in this matrix [7].

5.1 Utility matrix

This representation allows the visualization of users preferences for items. Specifically, in the utility matrix each value $v_{m,u}$ represents the degree of preference of user u for movie m. The initial utility matrix has been built with existing values, that are integers in a scale from 1 to 5, coming from the Netflix training data set from which they are given in the format described in Section 2.4. For most entries no explicit information is given about the user's preference for that specific movie. It is important to specify that a missing value does not correspond to a zero and, therefore, does not mean the user dislikes the movie, but just that he has not rated it yet. The aim of this project is to predict these unknown entries. In order to do this, for a given movie m the recommender searches for the most similar k movies already rated by user u, movies $x_1, x_2, ..., x_k$, and uses the value contained in $v_{x,u}$ to fill the unknown rating.
Since it has been very difficult to find movies from x_1 to x_k with a high similarity value that have a rating given by user u, only some predictions have been possible. Experiments have been done on multiple sets of movies, as exposed in Section 6.

5.2 Root-mean-square error

For each set of predicted ratings the respective root-mean-square error has been computed. This measure is frequently used to estimate the accuracy of predicted values respect to real ones.
When computing the RMSE, the first elements required are the *residuals*. Residuals are the difference between actual values, often denoted as y and predicted values, represented as $\hat{y}$. Therefore, for each prediction i the corresponding residual is the result of $\hat{y}_i - y$. Residuals can be positive or negative, as the predicted value can under or over estimates the actual one.

Finally, the RMSE value is computed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y)^2}{n}}$$

6. EXPERIMENTS

Multiple attempts of prediction making have been performed on the data set using the text-based recommender. In this paper just the most significant examples are presented.
To make a prediction for user u on movie m the k most similar movies have been taken under consideration $(x_1, x_2, ..., x_k)$. Starting from the one with higher similarity value, x_1, the algorithm looks for a rating value given from user u to that movie. If there is no such a rating, the same research is done on the next movie and so on, until x_k is reached. If there is no rating even for $x_{k,u}$, the prediction is not made.
Since the utility matrix is very sparse few predictions have been obtained for small values of k (this means that in those k most similar movies there are no other ratings from u). Because of this problem, incremental values of k, from 5 to 100, have been considered. Even with *k=100* not all the predictions are possible. This situation, in which the utility matrix is very sparse and there are too few ratings to make strong prediction, is called *cold start problem.*
To reinforce predictions a solution can be a hybrid recommender, in which the addition of other methods, *e.g.* the collaborative filtering, reinforces uncertain predictions and helps obtaining impossible ones. A hypothetical hybrid recommender is presented in Section 7 as a future work.
No movie after *k=100* is considered as safe for the prediction process, because after this value the similarity tends to go under the 0.2 threshold.

6.1 Predictions on random movies

As a first attempt 1.000.000 random predictions have been evaluated. For this experiment ratings have been extracted from the Netflix data set and redone. Results are exposed in Table 7.

Table 7: RMSE obtained for increasing values of *k*. 1,000,000 random predictions

K value	RMSE	Predictions number
5	1.211060142	30
10	1.188177052	119
15	1.188177052	119
25	1.188177052	119
50	1.188177052	119
100	1.161317788	261

The RMSE of these predictions varies from ≈ 1.16 to ≈ 1.21. The first result, with *k=5*, is based on only 30 predictions. For values of k from 10 to 50 the number of possible predictions remains fixed. This happens because even decreasing the accepted similarity value no ratings are found from the

same user for similar movies. Increasing k value to 100, more predictions are possible.
The RMSE obtained is a good result, considering that CineMatch, the original recommender algorithm used by Netflix, has an RMSE of 0.9525. However, with this approach very few predictions have been made. Besides, it is not possible to increase the k value any more, otherwise predictions would be deteriorated by an insufficient level of similarity.

6.2 Predictions on movies between 1995 and 2000

Analyzing random movies very few predictions have been possible, due to the lack in available ratings (cold start problem). The second experiment has been done on recent movies, released between 1995 and 2000, supposing to produce more predictions. Table 8 exposes results obtained after this experiment.

Table 8: RMSE obtained for increasing values of *k*. Predictions on movies between 1995 and 2000.

K value	RMSE	Predictions number
5	0.500000000	4
10	0.919866211	13
15	0.919866211	13
25	1.270977819	39
50	1.490189753	571
100	1.630340447	3715

A total amount of 12,498,607 predictions have been attempted. Unfortunately, for low values of k, very few predictions have been evaluated. It is possible to observe how the RMSE grows with increasing values of k. However, since the number of predictions increases as well, it is not possible to know if there is a real correlation or if very low RMSE values are obtained just by chance and not because of the higher text similarity.
In the case of *k=50* and *k=100* the number of predictions grows, up to 3715, but the respective RMSE is high. It is preferable to integrate the system with other recommendation approaches, especially when the similarity between movies has low values.

6.3 Predictions on movies after 2000

Analyzing very recent movies, a total amount of 21,866,263 predictions have been attempted. Ratings to be predicted have been taken from the data set, properly filtered in order to keep just movies released after 2000 (up to 2005, when the data set was released). Results are presented in Table 9.

Table 9: RMSE obtained for increasing values of *k*. Predictions on movies after 2000.

K value	RMSE	Predictions number
5	1.470736488	122
10	1.480754727	353
15	1.484989833	385
25	1.469693846	400
50	1.511202775	1082
100	1.427864589	2062

The number of predictions effectuated, as expected, increases with higher values of k. The RMSE is stable, with an average of ≈ 1.49. This result can definitely be improved utilizing hybrid approaches and adding more information to the content based approach, as explained in Section 4.3.

6.4 Considerations on experiments

Results emerged from all the experiments show that the text-based recommender is able to find similarities with an acceptable value of RMSE. However, there are some limitations:

- The cold start problem implies lots of difficulties in finding similar items rated by the same user. Often, in order to obtain a prediction, it is necessary to decrease the similarity threshold and consider items far from the selected one ($k \geq 50$).
- Even if in some experiments predictions have brought to good results, in some other cases the obtained RMSE is high, with an upper bound of ≈ 1.7, often related to high values of k. It would be better to decrease the threshold k and use alternative methods for recommendations when there are no high similar movies rated by the same user. A possible hybrid approach is presented in the next section.

7. FUTURE WORK

In this section we propose an hybrid approach for the recommender system. This approach is under implementation but it is not completed due to data processing problems. The hybrid approach enriches the recommender adding information about similarities coming from a user-user collaborative filtering. To complete this approach is necessary to obtain similarity between users; in particular, for each user u the aim is to find the list of n users most similar to u. Since the data set contains more than one hundred million ratings it has been impossible to compute the cosine similarity between couple of users on a single machine. The idea of computing the desired output in a distributed way is left as future work.
The hybrid recommender is needed to mitigate the cold start problem and allow more recommendations with higher quality. The method is exposed in Figure 1 and described after.

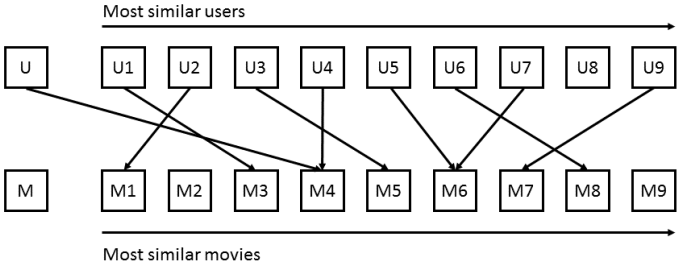

Figure 1: Hybrid approach for recommendations

For a prediction r made on the rating given by user u to movie m ($r_{m,u}$), the hybrid approach works as follow:

- for each user u a list of k most similar users is extracted;

- for each movie m a list of k most similar movies is extracted;
- decremental values (weight w_u) are given to all users, starting from u itself, to u_k;
- decremental values (weight w_m) are given to movies similar to m, starting from m_1 to m_k;
- scanning the two lists a reliability value is given to each rating r_i, computed as $w_{u,i} \cdot w_{m,i}$. The rating with higher reliability is chosen as suggestion.

With the hybrid approach suggestions are possible even for users that have rated few movies, even if the cold start problem can be just reduced and not completely eliminated.
Moreover, it is possible to obtain better predictions exploiting information from similar user activities, instead of searching for ratings given to movies with low cosine similarity values with respect to the target one.
As said in Section 4.3 other two enhancement to the content based approach are possible. First, it is suggested to include a list of proper nouns and delete them from texts as it is done with stop-words. Second, improvements can be achieved considering more features in content similarities. We are planning to include at least information about genre, actors, and directors.

8. CONCLUSIONS

Through this work it is possible to observe the effects of in-depth text analysis applied on a movie recommender system. Results show that text analysis can be considered a good feature to discover similarities between movies.
In particular, it has been observed that this approach is very good in finding episodes of series or alternative versions of a movie. These items are promoted with higher similarity values with respect to isolated movies. The approach works well in finding similar themes and contents among movies, giving them a more profound meaning when compared with simple keywords similarity approaches. Issues of this approach have also emerged, like the problem with characters names, that was not absolutely supposed at the beginning of our analysis.
Predictions effectuated using the text-based approach have acceptable RMSE values and, in some cases, good ones. However, due to the cold start problem, very few predictions are possible, thus depriving our work of strong empirical evidences.
An hybrid recommender can be a solution to decrease the cold start problem influence. We also believe that the quality of recommendations will increase with a proper combination of items representations.
Even if the results are satisfying, there are still some enhancement that can be applied to the content-based recommender, *e.g.* improvements of texts content and addition of more movie features.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender system. a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering (TKDE).*

[2] F. I. ans Y.O. Folajimib and B. Ojokohc. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal.*

[3] J. Bennett and S. Lanning. The netflix prize. *Proceedings of KDD Cup and Workshop 2007*, August 2007.

[4] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction archive.*

[5] C. B. G. Salton. Term-weighting approaches in automatic text retrieval. *Information Processing and Management Vol.24.*

[6] Y. Koren. The bellkor solution to the netflix grand prize. August 2009.

[7] J. Leskovec, A. Rajaraman, and J. Ullman. *Mining of Massive Datasets.*

[8] G. Linden, B. Smith, and J. York. Amazon.com recommendations: itemto-item collaborative filtering. *IEEE Internet Computing.*

[9] P. Lops, M. de Gemmis, and G. Seneraro. *Recommender systems handbook.*

[10] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *10th international conference on World Wide Web, Hong Kong*, May 2001.

[11] M. Uluyagmur, Z. Cataltepe, and E. Tayfur. Content-based movie recommendation using different feature sets. *WCECS 2012 - Proceedings of the World Congress on Engineering and Computer Science Vol I*, October 2012.