A non-intrusive movie recommendation system

A Non-intrusive Movie Recommendation System

Tania Farinella[1], Sonia Bergamaschi[2], and Laura Po[2]

[1] vfree.tv GmbH,
Agnes-Pockels-Bogen 1
80992 München, Germany
{tania.farinella}@vfree.tv,
http://vfree.tv/
[2] Department of Engineering "Enzo Ferrari",
University of Modena and Reggio Emilia,
Via Vignolese 905, 41125 Modena, Italy
{sonia.bergamaschi,laura.po}@unimore.it
http://www.dbgroup.unimo.it

Abstract. Several recommendation systems have been developed to support the user in choosing an interesting movie from multimedia repositories. The widely utilized collaborative-filtering systems focus on the analysis of user profiles or user ratings of the items. However, these systems decrease their performance at the start-up phase and due to privacy issues, when a user hides most of his personal data. On the other hand, content-based recommendation systems compare movie features to suggest similar multimedia contents; these systems are based on less invasive observations, however they find some difficulties to supply tailored suggestions.
In this paper, we propose a plot-based recommendation system, which is based upon an evaluation of similarity among the plot of a video that was watched by the user and a large amount of plots that is stored in a movie database. Since it is independent from the number of user ratings, it is able to propose famous and beloved movies as well as old or unheard movies/programs that are still strongly related to the content of the video the user has watched.
We experimented different methodologies to compare natural language descriptions of movies (plots) and evaluated the Latent Semantic Analysis (LSA) to be the superior one in supporting the selection of similar plots. In order to increase the efficiency of LSA, different models have been experimented and in the end, a recommendation system that is able to compare about two hundred thousands movie plots in less than a minute has been developed.

Keywords: Recommendation, Personalized Content, Movie, Latent Semantic Analysis

1 Introduction

Nowadays movie repositories offer datasets of over 100000 items and their size increases every year by around 5000 items due to the new released movies (according to Screen Digest[3]). Searching for a movie of interest in such a large amount of data is a time consuming task. *Information filtering* systems can be a powerful tool in giving assistance to the user. Thus, particularly in a multimedia environment, they are implemented to minimize user effort, to increase user satisfaction and to realize a more pleasant experience. For this purpose, recommendation methodologies have been integrated into customized media content distribution services. At the state of the art, the main methodologies analyze user profiles and user ratings of the data items to compute item similarity. Consequently, they find some difficulties from the start as user preferences are not necessarily available for the system. Moreover these systems are quite intrusive as they need active feed-back from users or their personal data. Content-based recommendation systems, instead, utilize movie features (such as title, director, year of production . . .) and, combining similarity measurements, they define how similar two movies are. While comparing movie features is quite easy, comparing plots is a challenging task; to our knowledge, none of the movie recommendation systems have proposed an algorithm based on the analysis of the plots till now. Moreover, the aim of this work was to offer recommendations that also include shows that are less popular or forgotten, because too old for example, but that can still be interesting for the user.

In this paper we propose a plot-based recommendation system which is based upon an evaluation of similarity among the plot of the movie that was watched by the user and a large amount of movie plots that is stored in a movie database. We exploit state-of-the-art text similarity techniques in order to evaluate similarity of natural language features such as plots. Then, we combined similarity of plots with similarity of non-verbose features, such as release year, crew etc. that are computed by exact matching.

In order to compare natural language features a vector space model was developed following the approach used in an Information Retrieval environment [8]. Within the vector space model, each text is represented as a vector of keywords with associated weights. These weights depend on the distribution of the keywords in the given training set of plots that are stored in the database. In order to calculate these weights we exploit and compare different techniques: simple weighting technique and semantic weighting techniques.

Weighting techniques such as Term Frequency-Inverse Document Frequency (tf-idf) and Log Entropy (log) assign a weight to each keyword that has been extracted from a text using lemmatizers and taggers. The vectors that are generated have a large size, as each of them consists of as many elements as many keywords have been extracted from the whole corpus of texts, and are very sparse, as all the keywords that do not appear in a text are associated in the corresponding vector to a zero-value element.

To generate small and non-sparse vectors (about 500 elements), the output of the cited weighting techniques is refined by applying LSA [14]. LSA allows to assign non-zero values to keywords that do not appear in a text but that are still related to its contents. The strong correlation between LSA weights

[3] http://www.screendigest.com/

and keyword co-occurrences allows to partially deal with synonymy (different keywords having similar meanings) and polysemy (keywords assuming different meanings).

The system has been developed in collaboration between the database group of the University of Modena and Reggio Emilia and vfree.tv[4], a young and innovative German company focused on creating new ways of distributing television content and generating an unprecedented watching experience for the user. Building upon several decades of experience of the founders in the fields of fixed and mobile telecommunication, video processing and distribution, the company is well equipped with a wealth of capabilities, indeed, in 2010 it won one of the five main awards of the German Federal Ministry of Economics and Technology for innovative new business ideas in the area of multimedia. Its products and services introduce a unique and disruptive technology for individual distribution of individual content. The user receives at any time the content which most likely satisfies his current needs and wants. vfree.tv works with service providers and content providers throughout Europe.

The paper is structured as follows. Section 2 introduces the structure of the local movie database that has been created. Section 3 describes the vector space model that has been used to compute similarity measures of natural language descriptions as plots. Section 4 compares the results obtained by tf-idf and LSA in selecting the 10 most similar movies to a given one. Moreover it examines the performance of approximated LSA models. Section 5 presents some related work, whereas conclusion and possible future evolvements are depicted in Section 6.

2 The Movie Database

With the aim of generating an extensive and reliable representation of multimedia, video metadata can be imported from external repositories and stored within a local database. Storing the data locally helps to improve the efficiency of processes that lead to the recommendation results. The local database should allow to easily enter data from different sources and perform queries on a huge amount of data (thousands of movies) in a short time and get good results. For these reasons we chose MongoDB.

MongoDB[5] is a non relational database and is schema-free, this feature allows to create databases with flexible and simple structure without decreasing the time performance when they are queried. Data is organized into collections, which correspond to tables in relational databases, and documents, the equivalent of tuples. MongoDB documents, as well as tuples, consist of a set of attributes, but since the database is schema-free the structure of each document is independent and potentially different from the structure of all the other documents in the same collection. It is then possible to change the structure of a document just modifying its attributes. MongoDB supports a query language

[4] http://vfree.tv

[5] http://www.mongodb.org/

that allows to define most of the queries that can be expressed in SQL. Furthermore, test demonstrated that for collection having a big size MongoDB shows better query performances [28].

In order to structure the local database, an analysis on the major movie repositories has been conducted. We took into consideration the Internet Movie Database (IMDb)[6], DBpedia[7] and the Open Movie Database (TMDb)[8]. Information about movies can be classified in either information that is related to multimedia or information that is about people that participated in the production of multimedia. This led to the creation of three main databases (as shown in Figure 1). The *Movie* database comprises data that is solely related to multimedia, such as title, plot and release year. The *Person* database comprises data related to people (e.g. full name, biography and date of birth). The *Castncrew* database connects documents of the other two databases. It comprises data about roles that are covered by people in the production of multimedia, such as actor, director or producer. This configuration allows an easier adaptation to integrate different/new datasets into the system, when external sources are shut down or experience a change in their copyright. Information extracted from different online resources can be stored separately as collections of the databases. For example as both IMDb and DBpedia supply information about movies and persons, they might potentially supply collections for all the three databases. However, if we have a repository that manages information about actors only, we can store this information adding a new collection in the database person. As MongoDB does not require a fixed schema, different collections in the databases may store different attributes. On the other hand, if we want to aggregate information about movies from different collections we connect information from the databases based on the name of the actors, the title of a movie, the year of production and other features if available.

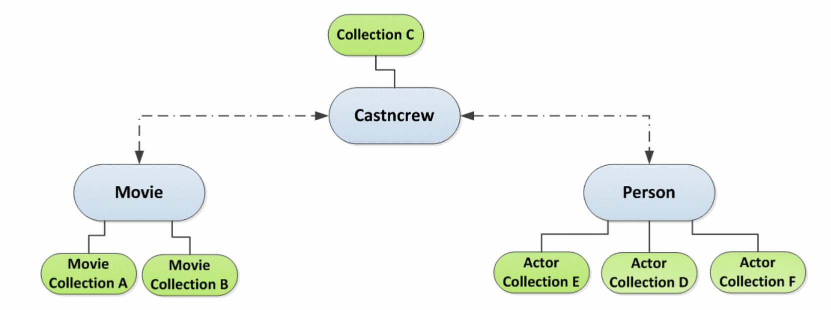

Fig. 1. The local MongoDB database

[6] http://www.imdb.com/
[7] http://dbpedia.org/
[8] http://www.themoviedb.org/

3 The Vector Space Model

The similarity of two media items depends on their features likeness. Hence, for each feature, a specific metric is defined in order to compute a similarity score. Most of the metrics that are adopted are calculated through only few simple operations. Things are more difficult for plots and, in general, for natural language descriptions. Our approach, that follows the one developed in an Information Retrieval environment [8], is based on a vector space model which is used to compare any pair of plots. Within this model, each text is represented as a vector of keywords with associated weights. These weights depend on the distribution of the keywords in the given training set of plots that are stored in the database. Vectors that represent plots are joined and consequently form a matrix where each row corresponds to a plot and each column corresponds to a keyword extracted from the training set descriptions. Thus, each cell of the matrix represents the weight of a specific keyword according to a specific plot. The weights in the matrix are determined in three steps. First, they are defined as the occurrences of keywords in the descriptions; second, weights are modified by optionally using the tf-idf or log technique (but other suitable weighting techniques could be used as well); third, the matrix is transformed by performing Latent Semantic Analysis (LSA) [12]. LSA is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. Several experiments have demonstrated that LSA has a good accuracy in simulating human judgments and behaviors [15].

3.1 Plot-based Similarity

Texts can be compared by using vector operations, such as the cosine similarity that is used as a distance metric in order to compute the similarity score between two texts, based on a vector representation.

Definition - cosine similarity *Given two vectors v_i, and v_j, that represent two different descriptions, the cosine angle between them can be calculated as follows:*

$$cosin(v_i, v_j) = \frac{\sum_k (v_i[k] \cdot v_j[k])}{\sqrt{\sum_k v_i[k]^2} \cdot \sqrt{\sum_k v_j[k]^2}}$$

The value of the cosine angle is a real number between -1 and 1. If the value is “1” the two compared vectors are equivalent, whereas if the value is “-1” the two vectors are opposite.

Thus, to compare descriptions by using vector operations plots need to be converted into vectors of keywords. As a preliminary operation, before the first step of our method, the keyword extraction and discrimination activity is performed. Keywords correspond to terms within the text that are representative of the text itself and that at the same time are discriminating. Less discriminative words, the so called *stop words*, are discarded and terms are preprocessed and substituted in the vector by their lemmas, this operation is called *lemmatization.* The goal of lemmatization is to reduce inflectional forms of a word to a common base form (e.g. to transfom "running", "runs" in the corresponding base form "run"). Lemmatization and keyword extraction are made by using *TreeTagger* [25]. TreeTagger is a parser that has been developed at the Institute for Computational Linguistics of the University of Stuttgart. This tool can annotate text with part-of-speech and lemma information in both the English and German language.

Keywords that have been extracted from descriptions as well as their local frequency (occurrences in the description) are stored as features of the media item in the local database. This happens for two main reasons. First, compared to the access to database values, the keyword extraction process is relatively slow[9]. As the weighting techniques define the values of the weights on the basis of the global distribution of the keywords over the whole corpus of descriptions, it is necessary to generate all the vectors before applying the tf-idf/log technique. Second, while tf-idf/log weights change when new multimedia descriptions are entered into the system, the local keyword occurrences do not.

Two different techniques have been used for computing keyword weights. In the following we briefly describe tf-idf technique, while we skip the definition of log. For major details we remand to [24] for an explanation of tf-idf and [9] for log. In both techniques, a weight that represents the relevance of a specific keyword according to a specific text depends on the local distribution of the keyword within the text as well as on the global distribution of the keyword in the whole corpus of descriptions.

Definition - tf-idf weight *Given a keyword k that has been extracted from a text the tf-idf weight is calculated as follows:*

$$weight_{tf-idf} = \frac{tf(k,d) \cdot idf(k)}{\sqrt{\sum_j v[j]^2}}$$

Where $tf(k,d)$ corresponds to the frequency of the keyword k in the vector v and $idf(k)$ depends on the number N of vector descriptions in the corpus and on the number df of vector descriptions in which the keyword k appears:

$$idf(k) = \log_2 \frac{N}{df}$$

tf-idf weights have a value between 0 and 1. Keywords with a document frequency equal to one are discarded.

The keyword weights are then refined by the use of LSA.

Let us introduce this technique by an example, suppose we have the following sentences *"There is a mouse below the new Ferrari that is parked in front of the market"*, *"With one mouse click you can view all available cars and thus renders going to the shop unnecessary"*. Now, let us compare the corresponding vectors that have been generated by tf-idf and LSA on the above sentences:

[9] One database access, using MongoDB, takes about 0.3 milliseconds while the extraction of keywords from a plot takes more than one second.

Table 1. A comparison of the weight vector correspondences obtained by using tf-idf and LSA

		ferrari	market	mouse	click	car	shop
tf-idf	v_1	0.67	0.38	0.48	0	**0**	**0**
	v_2	**0**	**0**	0.41	0.54	0.48	0.39
	product	**0**	**0**	>0	0	**0**	**0**
LSA	v_1	0.67	0.38	0.48	0	>**0**	>**0**
	v_2	>**0**	>**0**	0.41	0.54	0.48	0.39
	product	>**0**	>**0**	>0	0	>**0**	>**0**

The analysis of the values in Table 1 shows tf-idf is not able to recognize neither synonyms (e.g.*market* and *shop*) nor hyponyms (e.g. *Ferrari* and *cars*). In contrast, the use of LSA emphasizes underlying semantic: in the first sentence a value not equal to zero is assigned to the term *car* even if this keyword does not appear in vector v_1 (the vector that corresponds to the first sentence). This is due to the co-occurrences of the term *Ferrari* and other terms that also frequently occur in combination with the term car in other vectors of the training set matrix on which the LSA has been applied. There is a strong correlation between the values of the matrix and second order co-occurrences.

The LSA consists of a *Singular Value Decomposition* (SVD) of the vector matrix T (Training set matrix) followed by a *Rank lowering*. Each row and column of the resulting matrix T' can be represented as a vector combination of the eigenvectors of the matrix $T'T'^T$.

$$v = \sum_i coefficient_i \cdot eigenvector_i$$

Where the coefficients $coefficient_i$ of the above formula represent how strong the relationship between a keyword (or a description) and a topic $eigenvector_i$ is. The eigenvectors define the so-called *topic space*, thus, the coefficients related to a vector v represent a *topic vector*. Each eigenvector is referred to in the following as *topic*.

It is the topic representation of the keywords which is used as a natural language model in order to compare texts. Topic vectors may be useful for three main reasons: (1) as the number of topics that is equal to the number of non-zero eigenvectors is usually significantly lower than the number of keywords, the topic representation of the descriptions is more compact[10]; (2) the topic representation of the keywords makes possible to add movies that have been released after the definition of the matrix T' without recomputing the matrix T'; (3) to find similar movies starting from a given one, we just need to compute the topic vectors for the plot of the movie and then compare these vectors with the ones we have stored in the matrix T' finding the top relevant.

[10] Thus, we store the matrix of description-topic vectors to represent the training set.

Note that adopting this model we are able to represent each plot with 500 topics, instead of 220000 keywords.

3.2 Feature-based Similarity

The similarity of plots can also be combined with the similarity of other features such as directors, genre, producers, release year, cast etc. The similarity of two media items (m_1 and m_2) is defined as a weighted linear combination of the similarity of the feature values that describe the two items:

$$similarity(m_1, m_2) = \sum_{i=1}^{FN} weight_i \cdot similarity_i(feature_{1,i}, feature_{2,i})$$

Where FN is the number of features that describe a media item that has been chosen to compute the media similarity, $similarity_i$ is the metric used to compare the i-th feature, $feature_{j,k}$ is the value assumed by feature k in the j-th media item. The result of each metric is normalized to obtain a value between zero and one where one denotes equivalence of the values that have been compared and zero means maximum dissimilarity of the values [7].

4 Tests

In order to perform an evaluation of the developed recommendation system, we loaded data from IMDb for test purposes into the local database.

Table 2. A comparison between the tf-idf and log weighting techniques on the movie "The Godfather"

tf-idf	**log**
1. The Godfather	1. The Godfather
2. The Godfather Part II	2. The Godfather Part II
3. The Godfather Part III	3. Godfather Trilogy
4. The Godfather Saga	4. The Godfather Part III
5. Godfather Trilogy	5. Long Arm of the Godfather
6. The Rain People	6. Godfather
7. The Score	7. The Godfather Saga
8. Godfather	8. Mumbai Godfather
9. Three Godfathers	9. The Score
10. Mumbai Godfather	10.The Black Godfather

Most of the existing movie recommendation systems are based on collaborative filtering and build the movie proposal set analysing users ratings. Being famous programs more often rated, the proposed movies are usually the most famous ones. On the contrary, we are able to propose shows that are less popular or forgotten but that can still be interesting for the user. As this kind of results is new and there is not a single measure to compute similarity of multimedia plots, the evaluation of the results lacks an absolute benchmark. Anyway, results that we obtained, by using different techniques, are compared and briefly discussed.

Results obtained by applying tf-idf and log techniques on the local database show slight differences, and the quality does not seem to be significantly different. Instead, a noticeable quality improvement can be achieved by applying the LSA technique. Plots that are selected to be similar using tf-idf or log techniques usually contain terms and names that appear in the target plot, but they do not necessary refer to similar topics. LSA allows to select plots that are better related to the target's plot themes.

We report some manual tests that have been performed in order to evaluate and compare the weighting techniques. In Table 2 it can be noticed that the qualities of the results that could be achieved by the tf-idf and log techniques do not seem to be significantly different. In both of the ranked lists eight plots refer to the target plot whereas the other two plots seem to be related to similar content (godfathers). It is, therefore, not reasonable to suggest using one technique rather than the other.

Table 3. A comparison between the tf-idf technique application only and the further application of LSA on the movie "Inception"

LSA	**tf-idf**
1. Inception	1. Inception
2. The Dream Team with Annabelle and Michael	2. Cobb
3. Rainbow's Children	3. Somewhere in Georgia
4. In Pursuit of a Dream	4. Firecreek
5. Persistence of her Memory	5. House IV
6. Twenty-Seven Stories	6. Whiplash
7. The Silver Key	7. House
8. Mariposa china	8. Following
9. The Boat People	9. The Incident
10. Twee dromen	10. Lefty-Right

Table 3 shows the outcome of Latent Semantic Analysis is superior to other techniques such as tf-idf. Here, all plots that have been selected by using LSA technique refer to the theme of dreams and subconsciousness just like the target plot. In contrast, the results originating from tf-idf seem to be connected to the target plot more by the surname of the main character of the movie "Inception", which is Cobb.

Finally, in table 4 LSA results for the target movie series "Smallville" are compared, the evaluation took into consideration LSA over tf-idf and LSA over log techniques. All the plots that have been selected in both techniques refer to

Table 4. A comparison between the LSA over tf-idf and LSA over log weighting techniques on the movie "The Godfather"

LSA over tf-idf	**LSA over log**
1. Smallville	1. Smallville
2. Adventures of Superman	2. Adventures of Superman
3. The Stolen Costume	3. The Adventures of Superboy
4. Lois & Clark: The New Adventures ...	4. The Stolen Costume
5. Superman: The Animated Series	5. Legion of Super Heroes
6. Superman Returns	6. Superman: The Animated Series
7. Superman (series)	7. Panic in the Sky
8. Superboy	8. Lois & Clark: The New Adventures ...
9. Stamp Day for Superman	9. Lucy and Superman
10. Superman	10. All That Glitters

the topic of super-heroes and Superman. Just like in table 2, it is hardly possible to decide which results are of a better quality between Tf-idf and Log. The tf-idf technique might be preferred as, in contrast to the log technique, it does not require discarding terms that have a document frequency equal to one.

LSA process is based on the SVD of the plot-keyword matrix having a complexity of $O(m \times n)$ where m is the number of multimedia (rows of the matrix) and n is the number of keywords (columns) and $m \geq n$. There are about 200000 multimedia for which a plot value is available in the database after the IMDB data import and almost 220000 different keywords that are extracted from the plots. Thus, the time cost for the decomposition of the matrix is $O = 3 \cdot 10^{15}$. Furthermore, the decomposition requires random access to the matrix [5], which implies an intensive usage of the central memory. In order to efficiently compute the decomposition of the matrix and to avoid the central memory saturation, we utilized the framework Gensim[11].

The computational costs to create the LSA model on the local Linux system (4 AMD Phenom(tm) II X4 695 3.4 GHz processors, 3.6 GB RAM) are the following:

Given a target plot, all the other plots in the database can be ranked according to their similarity in about 42 seconds. To further decrease similarity time consumption, three LSA models have been built using different assumptions. These tests have been conducted on the data extracted from DBpedia. Table 6 summarises the time performance of our recommendation system obtained using the different models. The *complete model* includes all the movies having a plot (78602 movies) and all the keywords appearing in these plots (133369); in this model the matrix rank is reduced by LSA to 500 (LSA topics). While generating the *approximate model*, short plots (less than 20 keywords) and low-frequency keywords (appearing in less than 10 plots) have been ignored. Tf-idf and log weights having a value below 0.09 and LSA weights having a value below 0.001 have been set to 0. Here, the matrix rank has been reduced to 350. The *fast*

[11] http://radimrehurek.com/gensim/

Table 5. Computational Costs

Operation	**Description**	**Time**	**CPU average use**	**Central Memory average use**
Plot vectorization	Each plot is converted in a vector of keywords and corresponding frequencies	5 minutes	75%	11%
Tf-idf weights or Log wrights	For each keyword the tf-idf or the log weight is computed	1 minute	97%	10%
LSA weights	For each keyword the lsa weight is computed	9.5 hours	97%	42%

model is a further approximation of the approximate model in which the tf-idf and log weights having a value below 0.14 have been set to 0 and the matrix rank has been reduced to 200.

Table 6. Similarity time costs obtained using complete, approximate and fast models

	Complete model	Approximate model	Fast model
minimum document frequency	1	**10**	10
minimum vector length	1	**20**	20
minimum tf-idf weight	0	**0.09**	**0.14**
minimum log weight	0	**0.09**	**0.14**
minimum lsa weight	0	**0.001**	0.001
number of lsa topics	500	**350**	**200**
matrix size (rows x columns)	78602 x 133369	68038 x 15869	68038 x 15869
Similarity time cost	12 seconds	7 seconds	5 seconds

With the help of a different parametrization, it is thus possible to significantly cut down the time cost for computing similarity operations. Anyway, the more the model is approximated, the lower the accuracy becomes[12].

As described in section 3, the similarity of plots can be combined with the similarity of other features. We performed an experiment to compare the IMDb recommendation list for the movie "The Matrix" with the recommendations that have been generated by our system (we used the complete LSA model). In Table 7 are shown both the proposals that have been generated considering the plot only or plot together with other features.

[12] The three models have been tested performing the similarity ranking of the local database in regard to 10 different target-movies.

Table 7. A comparison on the different recommendation lists obtained for the movie "The Matrix"

IMDB Recom.		Plot-based Recom.		Feature-based Recom.	
Title	**Ratings**	**Title**	**Ratings**	**Title**	**Ratings**
The Matrix Reloaded	203557	Plug & Pray	50	The Matrix Reloaded	203557
The Matrix Revolutions	169963	Die Millennium-Katastrophe - Computer-Crash 2000	99	The Matrix Revolutions	169963
Star Wars: Episode V - The Empire Strikes Back	357393	Computer Warriors	24	Nezi: The Night of the Crazy Screws	0
Star Trek (2009)	205011	Colossus: The Forbin Project	3254	In the Realm of the Hackers	41
Rise of the Planet of the Apes	120325	The KGB, the Computer and Me	42	Plug & Pray	50
Outlander	28040	The Responsibilities of Men	0	The Unbearable Whiteness of Dean	7
The Time Machine	47159	In the Realm of the Hackers	41	The Quest for _____	15
Mad Max Beyond Thunderdome	34964	The Computer Virus	0	2010: A Kitchen Odyssey	9
Dune	48985	10. Le clone	12	Unauthorized Access	12

As it can be observed, the IMDb recommendation includes only movies that have been rated by a high number of users (more than 28000), while in the plot-based and in the feature-based recommendations even movies that are not famous have been proposed to the user. IMDB is not able to suggest movies similar to a selected plot. As a matter of fact, except for the movies of the Matrix trilogy, the IMDB recommendations are not related to the topic of "The Matrix", but they are rather the most popular science fiction movies. In contrast to the IMDB suggestions, both our algorithms are able to list a set of movies that are all related to at least one of the topics such as intelligent machines, computer hacker, computer viruses etc. An interesting observation is that, since the feature-based algorithm combines plots and features to select the recommendations, it is able to select the Matrix trilogy (that are of course the more related items) and a list of movies strongly related to the topic of "The Matrix".

5 Related Work

Information filtering systems can be a powerful tool in giving assistance to the user with the aim of delivering a narrow set of items which might be of interest. Recommendation algorithms are usually classified in content-based, collaborative filtering and hybrid recommendation systems [1]. Collaborative filtering systems are widely industrially utilized, for example by Amazon [18], MovieLens [19] and Netflix [4], and recommendation is computed by analysing user profiles and user ratings of the items. When user preferences are not available, as in the start-up phase, or not accessible, due to privacy issues, it might be necessary to develop a content-based recommendation algorithm, as the one proposed in [16]. Collaborative filtering and content-based approaches are combined in hybrid systems as in [2, 6]

Content-based recommendation systems rely on item descriptions that usually consist of punctual data. In [20] information is instead extracted by text to perform a categorization that supports the rating-based book recommendation. Herby we propose, instead, a recommendation approach that is based on natural language data, such as movie plots. Jinni[13] is a movie recommendation system that analyses as well movie plots, but relies on user ratings, manual annotations and machine learning techniques.

Descriptions of multimedia contents can be extracted from suitable data sources. [3] utilizes movie features that have been extracted from IMDB and [7] shows how the similarity of the features can be combined to define the distance of two movies, but none of these works involves the analysis of movie plots. Many efforts in other research areas, like schema matching and ontology matching, developed keyword similarity techniques exploiting lexical resources [27].

Evaluating the similarity among movies is closely related to the task of text similarity. Text similarity is essentially the problem of detecting and comparing the features of two texts. One of the earliest approaches to text similarity is

[13] http://www.jinni.com/

the vector-space model [24] with a term frequency / inverse document frequency (tf/idf) weighting. This model, along with the more sophisticated LSA semantic alternative [14] has been found to work well for tasks such as information retrieval and text classification. LSA was shown to perform better than the simpler word and n-gram feature vectors in an interesting study [17] where several types of vector similarity metrics (e.g., binary vs. count vectors, Jaccard vs. cosine vs. overlap distance measure, etc.) have been evaluated and compared.

In [16], Newsweeder, a news recommendation system is described, that relies on tf-idf. Beside weighting systems, mathematical techniques can be used to improve similarity results, such as LSA. Thanks to the analysis of word co-occurrences LSA allows to partially deal with the problem of polysemy and synonymy and to outperform weighting systems [17].

Due to the high computational cost of LSA there have been many work around in the area of approximate matrix factorization; these algorithms maintain the spirit of SVD but are much easier to compute [13]. For example, in [11] an effective distributed factorization algorithm based on stochastic gradient descent is shown. We opted for a scalable implementation of the process that does not require the term-document matrix to be stored in memory and is therefore independent of the corpus size [23].

6 Conclusions and Future Work

The paper presented a plot-based recommendation system. The system classifies two videos as being similar if their plots are alike. A local movie database with a flexible structure has been created to store a large amount of metadata related to multimedia content coming from different sources with heterogeneous schemata. Three techniques to compare plot similarity have been evaluated: tf-idf, log and LSA. From the results obtained, LSA turned out to be superior in supporting the selection of similar contents. Efficiency tests have been performed to speed up the process of LSA computation. The tests led to the development of a recommendation system able to compare the plot of a movie with a 200000 plots database. The results are provided in a ranked list of similar movies in less than one minute. An innovative feature of the system is its independence from the movie ratings expressed by users; this allows the system to find strongly related movies that other recommendation systems, such as IMDB, do not consider.

Keywords extraction might benefit from the use of lexical databases as WordNet [10] as they are particularly helpful in dealing with synonyms and polysemous terms. In WordNet, words (i.e. lemmas) are organized in groups of synonyms called synsets. Synsets are connected depending on semantic relationships such as hypernymy and hyponymy. Each keyword might be replaced by its meaning (synset), before the application of the weight techniques. To understand which of the synsets better express the meaning of a keyword in a plot we may adopt Word Sense Disambiguation techniques [21]. The semantic relationships between synsets can be used for enhancing the keyword meaning by adding all its hypernyms and hyponyms [22, 26].

In section 4, we performed some tests using IMDb data and other tests using DBpedia data; however we have not yet tested the system with a large amount of data coming from different sources. As a future work, we will evaluate other movie repositories, such as the Open Movie Database (TMDb) and the Rotten Tomatoes[14].

Acknowledgments. The work hereby presented has been possible thanks to the Erasmus internship of Tania Farinella at vfree.tv. Particular appreciation goes to Thomas Werner and Andreas Lahr[15], founders of vfree.it, for their suggestions and collaborations during the intership and the valuable comments on the paper.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17(6):734–749, 2005.
2. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.
3. C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.
4. J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.
5. M. Brand. Fast online svd revisions for lightweight recommender systems. In *In SIAM International Conference on Data Mining*, 2003.
6. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.
7. S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 1041–1042, New York, NY, USA, 2008. ACM.
8. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
9. S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236, 1991.
10. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, illustrated edition edition, May 1998.
11. R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 69–77, New York, NY, USA, 2011. ACM.

[14] http://developer.rottentomatoes.com/

[15] thomas.werner@vfree.tv, andreas.lahr@vfree.tv

12. A. Kontostathis and W. M. Pottenger. A framework for understanding latent semantic indexing (lsi) performance. *Inf. Process. Manage.*, 42:56–73, January 2006.
13. Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.
14. T. K. Landauer and S. T. Dutnais. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, pages 211–240, 1997.
15. T. K. Landauer, D. Laham, and P. Foltz. Learning Human-like Knowledge by Singular Value Decomposition: A Progress Report. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.
16. K. Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.
17. M. D. Lee and M. Welsh. An empirical evaluation of models of text document similarity. In *In CogSci2005*, pages 1254–1259. Erlbaum, 2005.
18. G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.
19. B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: Experiences with an occasionally connected recommender system. In *Proceedings of ACM 2003 Conference on Intelligent User Interfaces (IUI'03) (Accepted Poster)*, Chapel Hill, North Carolina, 2003. ACM.
20. R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.
21. R. Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), 2009.
22. L. Po and S. Sorrentino. Automatic generation of probabilistic relationships for improving schema matching. *Inf. Syst.*, 36(2):192–208, 2011.
23. R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.
24. G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.
25. H. Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, 1994.
26. S. Sorrentino, S. Bergamaschi, M. Gawinecki, and L. Po. Schema label normalization for improving schema matching. *Data Knowl. Eng.*, 69(12):1254–1273, 2010.
27. R. Trillo, L. Po, S. Ilarri, S. Bergamaschi, and E. Mena. Using semantic techniques to access web data. *Inf. Syst.*, 36(2):117–133, 2011.
28. G. Yunhua, S. Shu, and Z. Guansheng. Application of nosql database in web crawling. *International Journal of Digital Content Technology and its Applications (JDCTA)*, 5(6):261–266, 2011.