Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems

Sonia Bergamaschi and Laura Po(✉)

Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, 41125 Modena, Italy
{sonia.bergamaschi,laura.po}@unimore.it
http://www.dbgroup.unimo.it

Abstract. We propose a plot-based recommendation system, which is based upon an evaluation of similarity between the plot of a video that was watched by a user and a large amount of plots stored in a movie database. Our system is independent from the number of user ratings, thus 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. The system implements and compares the two Topic Models, Latent Semantic Allocation (LSA) and Latent Dirichlet Allocation (LDA), on a movie database of two hundred thousand plots that has been constructed by integrating different movie databases in a local NoSQL (MongoDB) DBMS. The topic models behaviour has been examined on the basis of standard metrics and user evaluations, performance assessments with 30 users to compare our tool with a commercial system have been conducted.

Keywords: Movie recommendation · LDA · LSA · Recommendation systems

1 Introduction

A recommendation system helps users to make choices without sufficient personal experience of all the possible alternatives [1]. These system are the basis of the targeted advertisements that account for most commercial sites revenues. They are commonly used on several fields, for suggesting entertainment items like books, music, videos and also for finding people on dating sites. Recommendation systems have become relatively successful at suggesting content, however their performance greatly suffers when little information about the user's preferences is given. These situations are not rare; they usually occur when the users are new to a system, the first time a system is launched on the market (no previous users have been logged), for new items (where we do not have any history on preferences yet) and when, because of user desire for privacy, the system does

© Springer International Publishing Switzerland 2015
V. Monfort and K.-H. Krempels (Eds.): WEBIST 2014, LNBIP 226, pp. 247–263, 2015.
DOI: 10.1007/978-3-319-27030-2_16

not record their preferences [2]. In such cases, making suggestions entirely based on the content that is being recommended can be a good solution.

In recent years, some events catalized the attention on movie recommendation systems: on 2009 and 2013 Netflix announced two developer contests for improving their predictions promising conspicuous prizes; on 2010 and 2011 two editions of the International Challenges on Context-Aware Movie Recommendation[1] took place.

The focus of our paper is to provide an automatic movie recommendation system that does not need any a priori information about users, i.e. a *completely non-intrusive system*. The paper compares two specific techniques (LDA and LSA) that have been implemented in our content-based recommendation system. Although topic models are not new in the area of recommendation systems, their use has not been deeper analyzed in a specific domain, such as the movie domain. Our intention is to show how these well-known techniques can be applied in this domain and how they perform.

The context where our system works is that of video-on-demand (VOD). Generally speaking, this is the case when a user is looking for an item without being registered on the site in which he is looking for (searching a book on Amazoon, a movie on IMDb etc.). We assumed that the only information we have about the user is his first choice, i.e. the movie he has selected/ he is watching. We do not have an history about his past selections nor a profile about his general interests. When watching a VOD movie, users explicitly request to buy and to pay for that movie, then what our system attempt to do is proposing a list of similar movies assuming that the chosen film has been appreciated by the user (the system assumes the user liked the movie if his play time is more then 3/4 of the movie play time). Here, we also assume that we have no knowledge about the preferences of a user; namely, about who is watching the film, and also profile about other users who have previously accessed the system.

There are dozens of movie recommendation engines on the web. Some require little or no input before they give you movie titles, while others want to find out exactly what your interests are. However all of these systems rely on ratings directly or indirectly expressed by the users of the system (some examples are *Netflix*, *Rotten Tomatoes*, *Movielens*, *IMDb*, *Jinni*).

Our movie recommendation system permits, given a movie, to supply the user with a list of those movies that are most similar to the target one. The way the system detects the list of similar movies is based upon an evaluation of similarity among the plot of the target movie and a large amount of plots stored in a movie database. This paper is an extended version of our previous work [3] and its main additions are the extended description of LSA and LDA models and an in-depth comparison of the use of LSA in contrast with LDA.

The paper is structured as follows. Section 2 describes the structure of the local movie database. Section 3 describes hoe the system performs the similarity computations among movie plots by using the LDA and LSA Topic Models. The experimental results of this study are presented in Sect. 4: we show the

[1] http://2011.camrachallenge.com/.

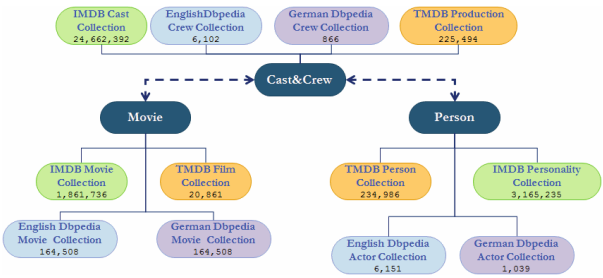

Fig. 1. The local MongoDB database.

computational costs of building the LSA and LDA matrices, and the results of off-line tests performed on three recommendation systems (LSA, LDA and a commercial approach). Section 5 presents some related work. Conclusion and future work are depicted in Sect. 6.

2 The Movie Database

The principal aim of a local repository of movies is to supply an extensive and reliable representation of multimedia that can be queried in a reasonable time. The local database of our system has been defined, as in [4], by importing data from external repositories. In particular, we selected the Internet Movie Database (IMDb)[2], DBpedia[3] and the Open Movie Database (TMDb)[4]. Since local database needs to easily import data from different sources and perform queries on a huge amount of data (thousands of movies) in a short time, we chose MongoDB[5], a non relational and schema-free database. MongoDB features allow to create databases with flexible and simple structure without decreasing the time performance when they are queried.

Information about movies can be classified in either information that are related to multimedia or information that are about people that participated in the production of multimedia. This led to the creation of three main databases, each storing collections from the 4 sources (as shown in Fig. 1). As MongoDB do not enforce document structure, this flexibility allows an easy adaptation to integrate different/new datasets into the system. A single collection can store documents with different fields. Thus there cannot really be a description of a collection, like the description of a table in the relational databases. An example of how these documents are organized within the database is shown in [3].

3 Plot Similarity Computation

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

[2] http://www.imdb.com/.
[3] http://dbpedia.org/.
[4] http://www.themoviedb.org/.
[5] http://www.mongodb.org/.

score. Most of the metrics that are adopted are calculated through only few simple operations. However, if we want to consider also movie plots, the similarity computation becomes more complex. Our approach is based on the Vector Space Model (VSM) [5], this model creates a space in which both documents and queries are represented by vectors. The VSM behaviour has also been studied applied on recommendation systems [6].

Our system takes advantage of this model to represent the different movie plots: each plot (or document from now on) 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 representing plots are then joined in a matrix representation where each row corresponds to a plot and each column corresponds to a keyword extracted from the training set plots (i.e. the *document by keyword matrix*). Thus, each cell of the matrix represents the weight of a specific keyword according to a specific plot.

The matrix computation goes through four main steps:

1. *Plot Vectorization* - relevant keywords are extracted and then stop words removal and lemmatization techniques are applied;
2. *Weights Computation* - weights are defined as the occurrences of keywords in the plots; the initial weights are then modified by using the tf-idf technique [5] (but other suitable weighting techniques could be used as well), thus building the *document by keyword matrix*;
3. *Matrix Reduction by using Topic Models* - the *document by keyword matrix* is reduced to a lower dimensional space by using the Topic Models LDA and LSA, thus it is transformed into a *document by topic matrix*.
4. *Movie Similarity Computation* - starting from the *document by topic matrix*, the similarity between two plots is computed by considering their topics as features instead of words.

3.1 Plot Vectorization

If two plots are to be compared, they will need to be converted into vectors of keywords. As preliminary operations, keyword extraction and filtering activity are performed. Keywords correspond to terms within the document that are representative of the document itself and that, at the same time, are discriminating. Less discriminative words, the so called *stop words*, are discarded, while the other terms are preprocessed and substituted in the vector by their lemmas (*lemmatization*). Lemmatization and keyword extraction are performed by using *TreeTagger*[6], developed at the Institute for Computational Linguistics of the University of Stuttgart. This tool can annotate documents with part-of-speech and lemma information in both English and German language. Keywords extracted from plots as well as their local frequencies (occurrences in the description of the plot) are stored as features of the media item in the local database

[6] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/.

MongoDB. This choice has been made for two main reasons. First, the keyword extraction process is relatively slow[7] compared to the access to database values. Since the weighting techniques are based on the global distribution of the keywords over the whole corpus of plots, it is necessary to generate all the vectors before applying the weighting technique. Second, while weights change when new multimedia plots are added into the system, the local keyword occurrences do not.

3.2 Weights Computation

Weighting techniques are used for computing keyword weights. A weight is a value in the range $[0, 1]$ that represents the relevance of a specific keyword according to a specific document. A weight is calculated on the basis of the local distribution of the keyword within the document as well as on the global distribution of the keyword in the whole corpus of plots. Keywords with a document frequency equal to 1 are discarded. Since, our previous work [4] has compared *tf-idf* and *log* weighting techniques revealing that the results are very similar, in this paper we employ only the tf-idf technique for computing the weights.

3.3 Matrix Reduction by Using Topic Model

The Vector Space Model treats keywords as independent entities. To find documents on specific concepts, we must provide the correct key terms. This representation leads to several issues: (1) there can be an high number of keywords when we have to deal with a huge amount of documents; (2) if any keyword changed, the document would not convey the same concept.

These problems can be faced by a representation into a Topic Model [7]. The Topic Model explores the idea that the concept held by a set of terms can be represented as a weighted distribution over a set of topics. Each topic is a linear combination of terms, where to each term a weight reflecting the relevance of the term for that topic is associated. For example, high weights for *family* and *house* would suggest that a topic refers to a social unit living together, whereas high weights for *users* and *communication* would suggest that a topic refers to social networking.

Topics can be found by clustering the set of keywords. The use of topics drastically reduces the dimension of the keyword Matrix obtained by Vector Space Model. Moreover, if a keyword changes, the document conveys the same idea as long as the new keyword is taken from the same topic pool.

Topic vectors may be useful in the context of movie recommendation systems for three main reasons: (1) 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 plots is more compact[8]; (2) the topic representation of the keywords makes possible to add movies that have been released after

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

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

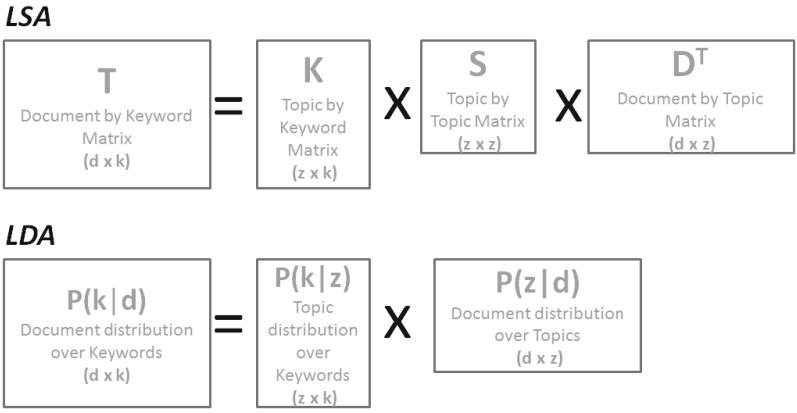

Fig. 2. Matrix decomposition for LSA and LDA.

the definition of the matrix without recomputing it; (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 finding the top relevant.

In the following, we describe the LSA and LDA models and how they can be applied to our movie recommendation system.

Latent Semantic Analysis (LSA). LSA is a model for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of documents. The LSA consists of a *Singular Value Decomposition* (SVD) of the matrix T (Training set matrix) followed by a *Rank lowering* [8]. The SVD consists of representing the matrix T as the product of three matrices (see also the top of Fig. 2):

$$T = KSD^T$$

Where K and D are orthogonal matrices and S is a diagonal matrix. The original matrix T is the *document by keyword matrix* as it represents the relationships between keywords and plots. Matrix K, *the topic by keyword matrix*, represents the relationships between keywords and topics. Matrix S is a diagonal matrix. Its values represent the square roots of the so called eigenvalues of the matrix TT^T. Matrix D^T, *document by topic matrix*, represents the relationships between plots and topics.

The SVD of the matrix is consequently followed by a *Rank lowering* by which the transformation of the matrix S into a matrix S' is performed which set the lowest values of S to zero. The purpose of dimensionality reduction is to reduce *noise* in the latent space, resulting in a richer word relationship structure that reveals latent semantics present in the collection. The T, S and D matrices are truncated to z dimensions (i.e. topics). The transformed *document by keyword matrix* T' is then obtained by the following product:

$$T' = KS'D^T$$

Each row and column of the matrix T' can be represented as a vector combination of the eigenvectors of the matrix $T'T'^T$

$$v = \sum_i c_i \cdot vector_i$$

Where the coefficients c_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*.

Queries are represented in the reduced space by $T_z^T q$, where T_z^T is the transpose of the term by topics matrix, after truncation to z topics. Queries are compared to the reduced document vectors, scaled by the singular values ($S_z D_z$) by computing the cosine similarity. The optimal z is determined empirically for each collection. In general, smaller z values are preferred when using LSA, due to the computational cost associated with the SVD algorithm, as well as the cost of storing and comparing large dimension vectors[9].

Latent Dirichlet Allocation (LDA). LSA provides a simple and efficient procedure for extracting a topic representation of the associations between terms from a term-document co-occurrence matrix. However, as shown in [9], this representation makes it difficult for LSA do deal with the polysemous terms. The key issue is that its representation does not explicitly identify the different senses of a term. To address this problem we investigated the use of the LDA Topic Model. LDA is a generative model for document collections that has been already successfully applied in several areas: document modeling and classification, Word Sense Disambiguation, Information Retrieval etc. [9–11].

Unlike LSA, LDA is a probabilistic Topic Model, where the goal is to decompose a conditional *term by the document probability distribution* $p(t|d)$ into two different distributions, *the term by topic distribution* $p(t|z)$, and *the topic by document distribution* $p(z|d)$ as follow (see also the bottom of Fig. 2):

$$p(k|d) = \sum_z p(k|z)p(z|d)$$

this allows each semantic topic z to be represented as a multinominal distribution of terms $p(k|z)$, and each document d to be represented as a multinominal distribution of semantic topics $p(z|d)$. The model introduces a conditional independence assumption that document d and keyword k are independent conditioned on the hidden variable, topic z.

In [9] it has been shown that LDA outperforms LSA, in the representation of ambiguous words and in a variety of other linguistic processing and memory tasks.

[9] In [4] work we determined 500 as a good number of topic. This value allows to have reasonable computational costs, and to maintain an appropriate level of accuracy.

LDA can be interpreted as matrix factorization where document over keyword probability distribution $p(k|d)$ can be split into two different distributions: the topic over keyword distribution $p(k|z)$, and the document over topic distribution $p(z|d)$. Thus, it appears clear, that we can easily make a direct correspondence between the document by topic matrix D^T obtained from LSA and the document over topic distribution $p(z|d)$ obtained by using LDA.

Both LDA and LSA permit to find a low dimensional representation for a set of documents w.r.t. the simple term by document matrix. This dimensionality in both cases has to be decided a priori. By adopting LSA, we were able to represent each plot of the IMDb database with 500 topics, instead of 220000 keywords[10]. For LDA (which has been demonstrated working well for a number of topics over 50 [10]), after a few experimental evaluations, we decided to use 50 topics.

3.4 Movie Similarity Computation

As previously described by using LSA or LDA the *document by keyword matrix* is decomposed into several matrices. The *document by topic matrix* (D^T or $P(z|d)$) is the one that is used to represent the movie of our database in a lower dimensional space, i.e. the matrix that is used to compute the similarity score between two plots.

The cosine similarity is used as a distance metric to calculate the similarity score between two documents. It is used to either compare plots within the training set or plots that are not included in the training set.

Definition - cosine similarity: *Given two vectors v_i, and v_j, that represent two different plots, 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 in the range $[-1, 1]$. If the cosine is equal to 1 the two vectors are equivalent, whereas if it is -1 the two vectors are opposite.

The similarity of plots can also be combined with the similarity of other features such as directors, genre, producers, release year, cast etc. as proposed in [4].

4 Experiments

We performed several tests in order to evaluate our system, the goal was to compare the effectiveness of LDA and LSA techniques and to evaluate the performance of the system on real users.

In [4] we have demonstrated that:

[10] Several experiments where conducted on a subset of the test set.

– There is not a big difference in the results obtained by applying log or tf-idf weighting techniques. Thus, we can use one of them.
– The use of the LSA model shows a noticeable quality improvement compared to the use of the SVD model. LSA allows to select plots that are better related to the target's plot themes.

Starting from these results, we conducted new tests and evaluations of the system. First of all, we loaded data from IMDb into the local database MongoDB and evaluated the computational costs of building the LSA and LDA matrices. Then, we compared the two Topic Models manually, by analyzing their behaviours in some special cases. Finally, we conducted off-line tests. We built two surveys asking real users to judge the similarity of each film in a list with regard to a target movie. The first test compared the performance of LDA and LSA. The second test compared the performance of LSA and a commercial system, IMDb. A third test evaluates the precision of the three recommendation systems.

4.1 Evaluation of the Computational Costs

The SVM of the plot-keyword matrix have a complexity of $O(d \times k)$ where d is the number of multimedia (rows of the matrix) and k is the number of keywords and $d \geq k$. There are about 1,861,736 multimedia in the IMDb database, but only for 200,000 there is a plot available. These plots contain almost 220,000 different keywords. 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, which implies an intensive usage of the central memory.

Both LSA and LDA decrease this cost by using a reduced matrix. The Document by Topic Matrix used by LSA has a dimension of $d \times z$ where z is the number of topic (columns). The Document Distribution over Topic Matrix used by LDA has a dimension of $d \times z$. Usually LSA requires more topics then LDA. Thus, the cost for the computation of the LDA matrix is further decreased. In order to avoid the central memory saturation, we employ the framework Gensim[11]. It offers functions to process plain document including algorithms performing both LSA and LDA which are independent from the training corpus size.

Table 1 shows the computational costs to create the LSA and LDA models[12].

Table 2 summarizes the configuration adopted for LSA and LDA and the time performance of the topic models when, starting from a given plot, they rank all the other plots in the database. Since the LDA model requires less topics (50 instead of the 500 required by LSA), it has a computation cost and a similarity time cost lower than the ones for LSA.

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

[12] The cost refers to a virtual machine set up with VMWare Workstation 9.0.1, installed on a server that has the following features: OS: Ubuntu 12.04 LTS 64-bit; RAM: 8 GB; 20 GB dedicated to the virtual hard disk; 4 cores. The DataBase Management System used is MongoDB 2.4.1, and it was installed on a machine with the following characteristics: OS: Windows Server 2008 R2 64-bit; CPU: Intel (R) Xeon E5620 Ghz 2:40; RAM: 12 GB.

Table 1. Computational costs.

Operation	Time (minutes)	CPU avg use	Memory avg use
Plot vect	5	75 %	11 %
Tf-idf weights	1	97 %	10 %
LSA weights	120	97 %	42 %
LDA weights	60	95 %	40 %

Table 2. LSA and LDA topic model comparison.

Configuration	LSA	LDA
min. document freq	10	10
min. vector length	20	20
min. tf-idf weight	0.09	0.09
min. lsa/lda weight	0.001	0.001
n. of topics	**500**	**50**
matrix size	204285×500	204285×50
Similarity time cost	**12 s**	**6 s**

4.2 Manual Comparison of Topic Models

We performed several manual evaluations in order to understand the behaviour of both the Topic Models. In Table 3 we report the top 5 recommendations calculated by using LDA and LSA models on two kind of movies: a movie of a saga and a movie that is not part of a sequel. For "The Matrix" (see the right part of the table), LSA selected movies referring to the topics of computer, network, programmer, hacker, while the outcome of the LDA technique showed two movies of the trilogy and other movies containing terms and names that appear also in the target plot, but that do not refer to similar topics. The quality of the outcome decreases with movies that do not have a sequel, like "Braveheart" (see the left part of the table). For this kind of movies, it is difficult to evaluate the recommended movie list. For this reason, we built a survey of popular movies that do not have a sequel and asked to real users to judge the similarity of the recommended movies (as presented in Sect. 4.3).

4.3 Testing the Recommendation System with Real Users

In order to evaluate the performance of our recommendation system, we identified two crucial steps: first it is necessary to understand which of the two Topic Models is more appropriate in the movie domain, then, we need to estimate its behaviour next to a commercial recommendation system, as IMDb. We defined three off-line tests: the first collecting the recommendations of LDA and LSA for 18 popular movies (excluding sagas), the second comparing the recommendations of the best Topic Model with respect to the recommended movie list of

Table 3. A comparison between LSA and LDA techniques.

	"Braveheart (1995)"		"The Matrix (1999)"	
	LSA	LDA	LSA	LDA
1	Braveheart (1995)	Braveheart (1995)	The Matrix (1999)	The Matrix (1999)
2	The Enemy Within (2010)	Windwalker (1981)	Computer Warriors (1990)	The Matrix Reloaded (2003)
3	Journey of a Story (2011)	Lipgloss Explosion (2001)	Electric Dreams (1984)	Simulacrum (2009)
4	Audition (2007)	Race for Glory (1989)	Willkommen in Babylon (1995)	Virus X (2010)
5	The Process (2011)	Voyager from the Unknown (1982)	TRON 2.0 (2003)	Fallen Moon (2011)
6	Comedy Central Roast of William Shatner (2006)	Elmo Saves Christmas (1996)	Hackers (1995)	The Matrix Revolutions (2003)

IMDb, the third analyzing in more detail the preferences expressed by 5 users on the three recommendation systems. We asked users to fill out the survey by selecting the films that looked similar to the film in question. These evaluations have enabled us to draw some conclusions on the performance of the implemented Topic Models and on our system in general.

LDA Versus LSA. The first off-line experiment involved 18 movies; for each of these movies, we selected the top 6 movies in the recommendation lists of both LSA and LDA. In order to propose results that can be easily judged by users, we discarded from the recommended movie lists: tv series, documentaries, short films, entries whose released year is before 1960, entries whose title is reported only in the original language, entries whose plot contains less than 200 characters.

We presented this list to users in a random order and asked them to judge for each movie in the list if it is similar to the target one, users can reply by choosing among "similar", "not similar" and "I do not know". Figure 4 reports the percentage of users' judgements received (here we do not consider the movies for which users have not expressed a judgement). We collected 594 evaluations from 20 users in total. We also evaluated the behavior of the Topic Models on each film: on the 18 movies, we found that in 15 cases LSA selected the best recommendations and in 3 cases LDA selected the best recommendations (see left part of Fig. 3). As expected from the previous comparison of the two models (reported in Table 3), LSA supplied better recommendations than LDA.

LSA Versus IMDb. In order to compare our system with respect to IMDb, we built another survey collecting recommendations for 18 popular movies (different with respect to the ones used in the LDA comparison): we selected them from

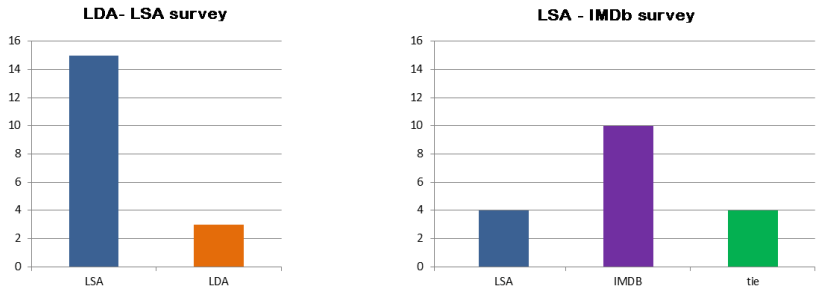

Fig. 3. Performance of the topic models and IMDb on the two surveys.

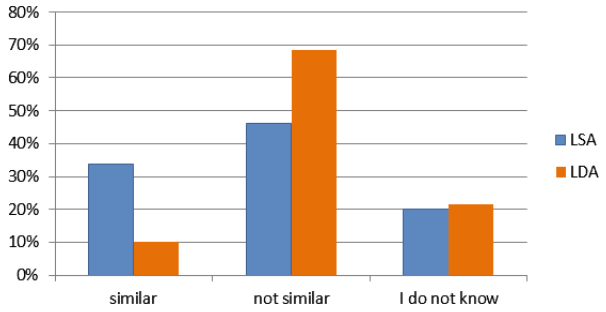

Fig. 4. Percentage of users' judgements on LSA-LDA survey.

the top 250 movies of IMDb[13]). Also in this case, we extracted only the top 6 movies in the recommendation lists of both LSA and IMDb.

In the previous survey, we obtained many void answers (i.e. on several recommended movies users do not expressed any opinion), moreover, some users highlighted that filling out the entire survey was very time consuming. Therefore, we decided to limit the options only to "similar".

We presented this list to users in a random order and asked users to judge for each movie in the list if it is similar to the target one. The experiment has been conducted on 30 test participants.

We collected 146 evaluations from 30 users in total. On the 18 movies, we found that in 4 cases LSA selected the best recommendations, in 10 cases IMDb selected the best recommendations and in 4 cases both systems showed the same performances (see right part of Fig. 3).

User Preference Evaluation. We added an in-deep evaluation of the users preferences for the 18 popular movies used in Sect. 4.3. This evaluation has been based on the precision measure computes by using the classification of recommendation results introduced in [12] as

[13] http://www.imdb.com/chart/top.

$$Precision = \frac{\#tp}{\#tp + \#fp}$$

On the 18 movies, we examine punctual preferences expressed by 5 expert users on the top 6 items of the recommendation list, for this evaluation we consider the "similar" and "not similar" judgement expressed by the users. Thus for each recommendation list we calculate the precision of the system based on the user judgement.

We computed the average precision among users (AVG_P@6) and the standard deviation among the movies (DEV_M_P@6) and the users (DEV_U_P@6) (see Table 4). AVG_P@6 reflects the average ratio of the number of relevant movies over the top-6 recommended movies for all users.

We found that the precision of LDA is quite low (about half as much as the LSA precision), while both LSA and IMDb reach a good precision. From this preliminary evaluation (that is quite limited since it is performed only on 5 users), it seems that the average precision on the entire set of movies of LSA is quite the same as the precision of IMDb. As it can be noticed, there is however a strong deviation of the precision value among different movies.

4.4 Results and Discussion

Based on the results of the above-mentioned experiments, we can draw some conclusions:

- LDA does not have good performance on movie recommendations as demonstrated by the user evaluation;
- LSA achieves good performance on movie recommendations;
- both LDA and LSA suggest erroneous entries for movies that have a short plot;
- our system did not outperform the IMDb performance; however, an in-deep evaluation of users preferences has shown that the average precision gained by LSA is very close to the precision of IMDb.

5 Related Work

Recommendation algorithms are usually classified in content-based and collaborative filtering [13]. Collaborative filtering systems are widely industrially utilized, for example by Amazon, MovieLens and Netflix, and recommendation is

Table 4. Precision of the systems based on a punctual user preference evaluation.

	AVG_P@6	DEV_M_P@6	DEV_U_P@6
LDA	0.215	0.163	0.133
LSA	0.468	0.258	0.056
IMDb	0.416	0.281	0.064

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, or combined different approaches as in hybrid systems.

Among recommendation systems [14], content-based recommendation systems rely on item descriptions that usually consist of punctual data. Jinni[14] is a movie recommendation system that analyses as well movie plots, but, differently from our approach, relies on user ratings, manual annotations and machine learning techniques.

LSA was shown to perform better than the simpler word and n-gram feature vectors in an interesting study [15] 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. 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 [16]. For example, in [17] 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 [18].

Also the LDA Topic Model has been already applied in recommendation systems to analyze textual information. In particular in [19] a Web recommendation system to help users in locating information on the Web is proposed. In this system LDA is used as technique for discovering hidden semantic relationships among Web items by analyzing their content information. Another interesting application is described in [20] where the authors propose a tag recommendation system where LDA is exploited to suggest tags for new resources.

In the specific domain of movie recommendation systems, we found only few frameworks that make use of plots. In particular in [21] a Context-Aware Recommendation algorithm is introduced, the framework combines the similarity based on plot keywords with a mood-specific movie similarity for providing recommendations. Also in [22] authors attempts to solve the cold start problem (where there is no past rating for an item) for collaborative filtering recommendation systems. The paper describes a framework, based on an extended version of LDA, able to take into account item-related emotions, extracted from the movie plots, and semantic data, inferred from movie features.

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. Two Topic Models, LDA and LSA, have been implemented and integrated within the recommendation system. The techniques have been compared and tested over a large collection of movies. The local movie database MongoDB has been created to store a large amount

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

of metadata related to multimedia content coming from different sources with heterogeneous schemata.

Experimental evaluation of both LDA and LSA has been conducted to provide answers in term of efficiency and effectiveness. LSA turns out to be superior to LDA. The performance of both the techniques have been compared to user evaluation, and commercial approaches. LSA has been revealed to be better then LDA in supporting the suggestion of similar plots, but it does not outperform the commercial approach (IMDb). However, we can not ignore that IMDb is strongly affected by user experiences: it uses features such as user votes, genre, title, keywords, and, most importantly, user recommendations themselves to generate an automatic response. On the contrary, our content-based recommendations system is user independent and can be also used to make recommendations when knowledge of user preferences is not available.

The results shown in this paper highlight some limitations and stimulate some future directions for our research: (1) *Combination with other techniques*, (2) *Making Use of Lexical Resources* and (3) *Exploring new application scenarios*.

(1) The plot-based recommendation techniques assume that the main feature a user likes in a movie is the plot, i.e. the content of the movie, if this is not the case, the system will fail in suggesting similar movies. An improvement might be to couple our recommendation system with other techniques that do not totally rely on the plot.

(2) While LDA deals with the polysemy issue, LSA does not. This problem can be faced by making use of a lexical database as WordNet[15]. 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 might adopt Word Sense Disambiguation techniques [23]. The semantic relationships between synsets can be used for enhancing the keyword meaning by adding all its hypernyms and hyponyms [24–26].

(3) Our system does not rely on human effort and can be ported to any domain where natural language descriptions exist (like news, book plots, book reviews etc.). For example, the system might be able to find movies that contain a story similar to the one told in a book, e.g. a movie or a television series that used it as a script, or dramatic movies based on true events similar to a news. The database could be expanded with other contents to suggest further items similar to the selected movie (e.g. if I liked a movie about the war in Cambodia I should be interested in newspaper articles, essays, or books about that topic).

Acknowledgements. The system has been developed in collaboration between the database group of the University of Modena and Reggio Emilia and vfree.tv (http://vfree.tv), a young and innovative German company focused on creating new ways of distributing television content and generating an unprecedented watching experience for the user.

We also want to express our gratitude to Tania Farinella, Matteo Abbruzzo and Olga Kryukova, master students in Computer Engineering and Science at the

[15] http://wordnet.princeton.edu/.

Department of Engineering "Enzo Ferrari" at University of Modena and Reggio Emilia for their contribution in term of implementation of the first and second version of the system (without and with LDA) and for their support during the evaluation of the system. Particular appreciation goes to Serena Sorrentino that helps us to integrate the LDA models in our system.

References

1. Resnick, P., Varian, H.R.: Recommender systems. Commun. ACM **40**, 56–58 (1997)
2. Rashid, A.M., Karypis, G., Riedl, J.: Learning preferences of new users in recommender systems: an information theoretic approach. SIGKDD Explor. Newsl. **10**, 90–100 (2008)
3. Bergamaschi, S., Po, L., Sorrentino, S.: Comparing topic models for a movie recommendation system. In: Proceedings of 10th International Conference on Web Information Systems and Technologies (WEBIST 2014), Barcelona, Spain, Number 2, SCITEPRESS, pp. 172-183 (2014). ISBN 978-989-758-024-6
4. Farinella, T., Bergamaschi, S., Po, L.: A non-intrusive movie recommendation system. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 736–751. Springer, Heidelberg (2012)
5. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**, 613–620 (1975)
6. Musto, C.: Enhanced vector space models for content-based recommender systems. In: Proceedings of the Fourth ACM Conference on Recommender Systems. RecSys 2010, pp. 361–364. ACM, New York (2010)
7. Park, L.A.F., Ramamohanarao, K.: An analysis of latent semantic term self-correlation. ACM Trans. Inf. Syst. **27**, 8:1–8:35 (2009)
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**, 391–407 (1990)
9. Griffiths, T., Steyvers, M., Tenenbaum, J.: Topics in semantic representation. Psychol. Rev. **114**, 211–244 (2007)
10. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
11. Sorrentino, S., Bergamaschi, S., Parmiggiani, E.: A supervised method for lexical annotation of schema labels based on wikipedia. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 359–368. Springer, Heidelberg (2012)
12. Gunawardana, A., Shani, G.: A survey of accuracy evaluation metrics of recommendation tasks. J. Mach. Learn. Res. **10**, 2935–2962 (2009)
13. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Found. Trends Hum.-Comput. Interact. **4**, 81–173 (2011)
14. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**, 734–749 (2005)
15. Lee, M. D., Welsh, M.: An empirical evaluation of models of text document similarity. In: Proceedings of the 27th Annual Conference of the Cognitive Science Society. CogSci2005, Erlbaum (2005) 1254–1259
16. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**, 30–37 (2009)

17. Gemulla, R., Nijkamp, E., Haas, P.J., Sismanis, Y.: 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 2011, pp. 69–77. ACM, New York (2011)
18. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, Valletta, Malta, ELRA, pp. 45–50 (2010)
19. Jin, X., Mobasher, B., Zhou, Y.: A web recommendation system based on maximum entropy. In: ITCC, pp. 213–218. IEEE Computer Society (2005)
20. Krestel, R., Fankhauser, P., Nejdl, W.: Latent dirichlet allocation for tag recommendation. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., Felfernig, A., Schmidt-Thieme, L. (eds.) RecSys, pp. 61–68. ACM (2009)
21. Shi, Y., Larson, M., Hanjalic, A.: Mining contextual movie similarity with matrix factorization for context-aware recommendation. ACM Trans. Intell. Syst. Technol. **4**, 16:1–16:19 (2013)
22. Moshfeghi, Y., Piwowarski, B., Jose, J. M.: Handling data sparsity in collaborative filtering using emotion and semantic based features. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR 2011, pp. 625–634. ACM, New York (2011)
23. Navigli, R.: Word sense disambiguation: a survey. ACM Comput. Surv. **41**, 1–69 (2009)
24. Po, L., Sorrentino, S.: Automatic generation of probabilistic relationships for improving schema matching. Inf. Syst. **36**, 192–208 (2011)
25. Sorrentino, S., Bergamaschi, S., Gawinecki, M., Po, L.: Schema label normalization for improving schema matching. Data Knowl. Eng. **69**, 1254–1273 (2010)
26. Bergamaschi, S., Bouquet, P., Giacomuzzi, D., Guerra, F., Po, L., Vincini, M.: An incremental method for the lexical annotation of domain ontologies. Int. J. Semantic Web Inf. Syst. **3**, 57–80 (2007)