Comparing Topic Models for a Movie Recommendation System

Sonia Bergamaschi, Laura Po and Serena Sorrentino
DIEF, University of Modena and Reggio Emilia, 41125 Modena, Italy

Keywords: Movie Recommendation System, LDA, LSA.

Abstract: Recommendation systems have become successful at suggesting content that are likely to be of interest to the user, however their performance greatly suffers when little information about the users preferences are given. In this paper we propose an automated movie recommendation system based on the similarity of movie: given a target movie selected by the user, the goal of the system is to provide a list of those movies that are most similar to the target one, without knowing any user preferences. The Topic Models of Latent Semantic Allocation (LSA) and Latent Dirichlet Allocation (LDA) have been applied and extensively compared on a movie database of two hundred thousand plots. Experiments are an important part of the paper; we examined the topic models behaviour based on standard metrics and on user evaluations, we have conducted performance assessments with 30 users to compare our approach with a commercial system. The outcome was that the performance of LSA was superior to that of LDA in supporting the selection of similar plots. Even if our system does not outperform commercial systems, it does not rely on human effort, thus it can be ported to any domain where natural language descriptions exist. Since it is independent from the number of user ratings, it is able to suggest famous movies as well as old or unheard movies that are still strongly related to the content of the video the user has watched.

1 INTRODUCTION

Recommendation systems are information filtering systems that recommend products available in e-shops, entertainment items (books music, videos, Video on Demand, books, news, images, events etc.) or people (e.g. on dating sites) that are likely to be of interest to the user. These system are the basis of the targeted advertisements that account for most commercial sites revenues. In the recent years, some events catalized the attention on movie recommendation systems: in 2009, a million-dollar prize has been offered by the DVD rental site Netflix[1] to anyone who could improve their predictions by 10%[2]; in 2010 and 2011 we saw the International Challenges on Context-Aware Movie Recommendation[3]; moreover, in 2013, Netflix announced a new developer contest called the "Netflix Cloud Prize"[4] which is promising a prize money of $100,000 to those who improve the performance of Netflix's cloud computing services.

Recommendation systems have become relatively successful at suggesting content, however their performance suffers greatly 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) (Adomavicius and Tuzhilin, 2005) and when, because of user desires for privacy, the system does not record their preferences (Rashid et al., 2008). In such cases, making suggestions entirely based on the content that is being recommended can be a good solution.

The focus of the paper is to provide an automatic movie recommendation system that does not need any a priori information about users. 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 such specific domain and how they perform.

The automatic movie recommendation system

[1] http://www.netflixprize.com/

[2] The grand prize was given to the BellKor's Pragmatic Chaos team which bested Netflix's own algorithm for predicting ratings by 10.06%.

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

[4] https://github.com/Netflix/Cloud-Prize/wiki

Bergamaschi S., Po L. and Sorrentino S..
Comparing Topic Models for a Movie Recommendation System.
DOI: 10.5220/0004835601720183
In *Proceedings of the 10th International Conference on Web Information Systems and Technologies* (WEBIST-2014), pages 172-183
ISBN: 978-989-758-024-6

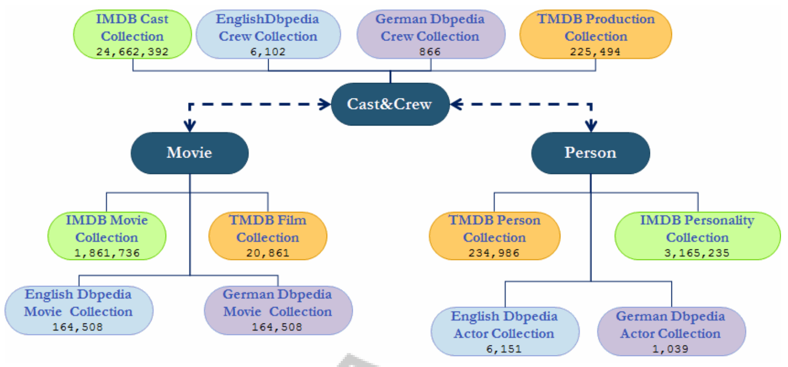

IMDB Cast Collection 24,662,392
EnglishDbpedia Crew Collection 6,102
German Dbpedia Crew Collection 866
TMDB Production Collection 225,494
Cast&Crew
Movie
Person
IMDB Movie Collection 1,861,736
TMDB Film Collection 20,861
English Dbpedia Movie Collection 164,508
German Dbpedia Movie Collection 164,508
TMDB Person Collection 234,986
IMDB Personality Collection 3,165,235
English Dbpedia Actor Collection 6,151
German Dbpedia Actor Collection 1,039

Figure 1: The local MongoDB database.

permits, given a movie, to supply users 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 that is stored in a movie database. The movie database has been constructed by integrating different movie databases in a local NoSQL (MongoDB) database, building a collection of about two hundred thousand plots together with the most relevant movie metadata.

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 the only information we have about the user is his first choice, the movie he has selected/ he is watching (we do not have a 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 the users; namely, about who is watching the film, and also with regard to 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 users of the system (some examples are *Netflix*, *Rotten Tomatoes*, *Movielens*, *IMDb*, *Jinni*).

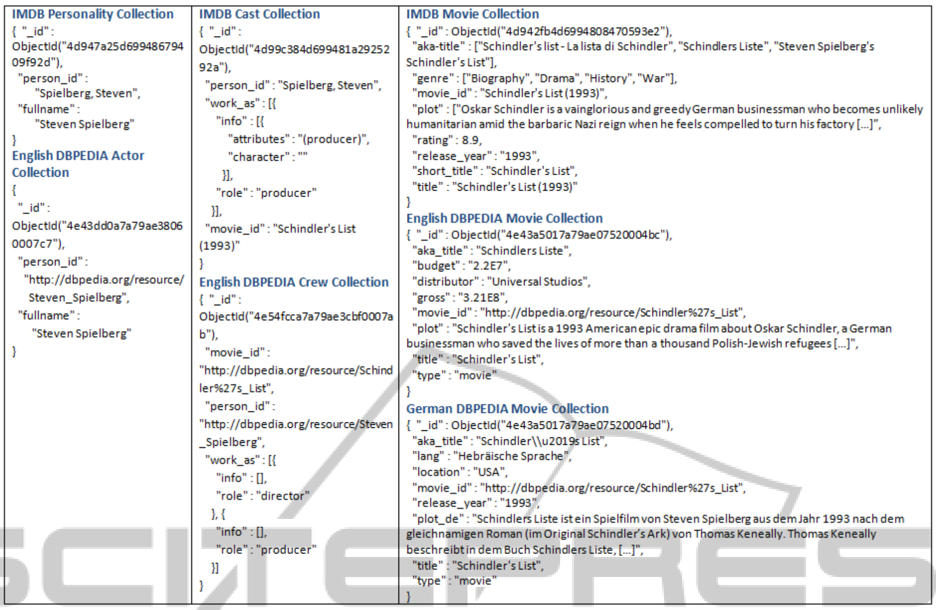
Starting from our previous work (Farinella et al., 2012) that highlighted the power of using the Latent Semantic Analysis (LSA) in contrast with weighting techniques (such as log and tf-idf) in suggesting similar movies, in this paper we intend to integrate in the system the Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and then, by evaluating on real users we compare the performance of our system based on LSA and LDA.

In (Griffiths et al., 2007) it has been shown that the Latent Dirichlet Allocation (LDA) Topic Model (Blei et al., 2003) outperforms LSA, in the representation of ambiguous words and in a variety of other linguistic processing and memory tasks. Hereby, we intend to analyze the performance of LDA on the movie domain and compare its behaviour with regard to LDA. In the end, we examine the performance of our system with regard to a commercial approach.

The system has been developed in collaboration between the database group of the University of Modena and Reggio Emilia[5] and vfree.tv[6], a young and innovative German company focused on creating new ways of distributing television content and generating an unprecedented watching experience for the user.

The paper is structured as follows. Section 2 describes the structure of the local movie database MongoDB. 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 Section 4: we show the 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 Section 6.

[5]http://www.dbgroup.unimo.it
[6]http://vfree.tv



IMDB Personality Collection

```
{ "_id" :
ObjectId("4d947a25d69948679409f92d"),
  "person_id" :
      "Spielberg, Steven",
  "fullname" :
      "Steven Spielberg"
}
```

English DBPEDIA Actor Collection

```
{
  "_id" :
ObjectId("4e43dd0a7a79ae38060007c7"),
  "person_id" :
    "http://dbpedia.org/resource/
     Steven_Spielberg",
  "fullname" :
     "Steven Spielberg"
}
```

IMDB Cast Collection

```
{ "_id" :
ObjectId("4d99c384d699481a2925292a"),
  "person_id" : "Spielberg, Steven",
  "work_as" : [{
     "info" : [{
        "attributes" : "(producer)",
        "character" : ""
       }],
     "role" : "producer"
    }],
  "movie_id" : "Schindler's List
(1993)"
}
```

English DBPEDIA Crew Collection

```
{ "_id" :
ObjectId("4e54fcca7a79ae3cbf0007ab"),
  "movie_id" :
"http://dbpedia.org/resource/Schindler%27s_List",
  "person_id" :
"http://dbpedia.org/resource/Steven_Spielberg",
  "work_as" : [{
     "info" : [],
     "role" : "director"
    }, {
     "info" : [],
     "role" : "producer"
    }]
}
```

IMDB Movie Collection

```
{ "_id" : ObjectId("4d942fb4d6994808470593e2"),
  "aka-title" : ["Schindler's list - La lista di Schindler", "Schindlers Liste", "Steven Spielberg's
Schindler's List"],
  "genre" : ["Biography", "Drama", "History", "War"],
  "movie_id" : "Schindler's List (1993)",
  "plot" : ["Oskar Schindler is a vainglorious and greedy German businessman who becomes unlikely
humanitarian amid the barbaric Nazi reign when he feels compelled to turn his factory […]",
  "rating" : 8.9,
  "release_year" : "1993",
  "short_title" : "Schindler's List",
  "title" : "Schindler's List (1993)"
}
```

English DBPEDIA Movie Collection

```
{ "_id" : ObjectId("4e43a5017a79ae07520004bc"),
  "aka_title" : "Schindlers Liste",
  "budget" : "2.2E7",
  "distributor" : "Universal Studios",
  "gross" : "3.21E8",
  "movie_id" : "http://dbpedia.org/resource/Schindler%27s_List",
  "plot" : "Schindler's List is a 1993 American epic drama film about Oskar Schindler, a German
businessman who saved the lives of more than a thousand Polish-Jewish refugees […]",
  "title" : "Schindler's List",
  "type" : "movie"
}
```

German DBPEDIA Movie Collection

```
{ "_id" : ObjectId("4e43a5017a79ae07520004bd"),
  "aka_title" : "Schindler\\u2019s List",
  "lang" : "Hebräische Sprache",
  "location" : "USA",
  "movie_id" : "http://dbpedia.org/resource/Schindler%27s_List",
  "release_year" : "1993",
  "plot_de" : "Schindlers Liste ist ein Spielfilm von Steven Spielberg aus dem Jahr 1993 nach dem
gleichnamigen Roman (im Original Schindler's Ark) von Thomas Keneally. Thomas Keneally
beschreibt in dem Buch Schindlers Liste, […]",
  "title" : "Schindler's List",
  "type" : "movie"
}
```

Figure 2: Documents related to the “Schindler’s List” movie.

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 our previous work (Farinella et al., 2012), by importing data from external repositories. In particular, we selected the Internet Movie Database (IMDb)[7], DBpedia[8] and the Open Movie Database (TMDb)[9]. 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[10], a non relational database and schema-free. 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 Figure 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. However, to give an insight into the local DB, we extracted some documents that store information related to the “Schindler’s List” movie. In figure 2, documents from different collections (IMDb, and the English and German version of DBpedia) are shown. It can be noticed how the information are heterogeneously represented in the collections (see the different information stored in the English and German version of DBpedia Movie collection) and how flexible the structure of each documents is (see for example in the English DBpedia Crew Collection the double role of Steven Spielberg that is both the director and the producer of the movie).

[7]http://www.imdb.com/
[8]http://dbpedia.org/
[9]http://www.themoviedb.org/
[10]http://www.mongodb.org/

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 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) (Salton et al., 1975), this model creates a space in which both documents and queries are represented by vectors. In the area of the Information Retrieval the VSM has been considered as one of the most effective approaches, and its behaviour has also been studied applied on recommendation systems (Musto, 2010).

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 (Salton et al., 1975)(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*[11], 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 MongoDB. This choice has been made for two main reasons. First, the keyword extraction process is relatively slow[12] 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 (Farinella et al., 2012) 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(Park and Ramamohanarao, 2009).

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

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

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[13]; (2) the topic representation of the keywords makes possible to add movies that have been released after 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.

The main Topic Models exploited so far in literature are the LSA (also called Latent Semantic Indexing (LSI)) and the LDA. In the following, we briefly describe both the methods and how they can be applied to our movie recommendation system.

3.3.1 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* (for more details see (Dumais, 2004; Deerwester et al., 1990)). The Singular Value Decomposition consists of representing the matrix T, the *document by keyword matrix* that represents the relationships between keywords and plots, as the product of three matrices: K, S, D^T. Matrix K, *the topic by keyword matrix*, represents the relationships between keywords and topics, while matrix S is a diagonal matrix whose 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 Singular Value Decomposition is consequently followed by a *Rank lowering* by which the matrices S and T are transformed respectively into the matrices S',T'. 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. In a LSA system, the matrices are truncated to z dimensions (i.e. topics). 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[14].

3.3.2 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 (Griffiths et al., 2007), 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 Latent Dirichlet Allocation (LDA) Topic Model.

Unlike LSA, LDA is a probabilistic Topic Model, where the goal is to decompose a conditional *term by document probability distribution* into two different distributions, this allows each semantic topic z to be represented as a multinominal distribution of terms, and each document d to be represented as a multinominal distribution of semantic topics. The model introduces a conditional independence assumption that document d and keyword k are independent conditioned on the hidden variable, topic z.

LDA can also be interpreted as matrix factorization where document over keyword probability distribution can be split into two different distributions: the topic over keyword distribution, and the document over topic distribution. Thus, it appears clear, that we can easily make a direct correspondence between the document by topic matrix obtained from LSA and the the document over topic distribution obtained by using LDA.

Both LDA and LSA permit to find a low dimensional representation for a set of documents with re-

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

[14]In our previous work we determined 500 as a good number of topic. This value allows have reasonable computational costs, and maintains an appropriate level of accuracy.

gard to 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 220,000 keywords (Farinella et al., 2012). For LDA (which has been demonstrated working well for a number of topics over 50 (Blei et al., 2003)), 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* is the one that is used to represent the movie of our database in a lower dimensional space and also to compute the similarity score between two plots.

To calculate the similarity score between two documents we use the cosine similarity. This metric 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.

Definition - Feature-based Similarity: *Given two media items (m_1 and m_2) the feature-based similarity is defined as a weighted linear combination of the similarity of the feature values that describe the two items:*

$$sim(m_1, m_2) = \sum_{i=1}^{FN} w_i \cdot sim_i(f_{1,i}, f_{2,i})$$

where FN is the number of features that describe a media item, sim_i is the metric used to compare the i-th feature, $f_{j,k}$ is the value assumed by feature k in the j-th media item. The result of each metric is normalized to obtain a value in the range $[0, 1]$ where 1 denotes equivalence of the values that have been compared and 0 means maximum dissimilarity of the values (Debnath et al., 2008).

4 EXPERIMENTS

We have 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.

Our previous research (Farinella et al., 2012) has shown that:

- 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 Topic Model LSA 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 Setup of the Environment

The DataBase Management System used is MongoDB 2.4.1, the system has been 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. The 64bit version of MongoDB is necessary as it is the only version that allow working with databases greater than 2 GB.

A virtual machine for the execution of the code has been installed on the server. The virtual machine 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 virtual machine has been set up with VMWare Workstation 9.0.1[15]. The 32-bit architecture makes it possible to instantiate a maximum of 2 GB of memory for a process. The creation of the LSA and LDA models exceeds the threshold of 2 GB, then the use of a 64-bit architecture is crucial in order to avoid memory errors.

[15]http://www.vmware.com/products/workstation/

4.2 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[16]. 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 (the cost refers to the environment that we described in 4.1).

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 x 500	204285 x 50
Similarity time cost	**12 sec**	**6 sec**

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.

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

Table 3: A comparison between LSA and LDA techniques on the movie "Batman Begins".

LSA	**LDA**
1.Batman Begins(2005)	1.Batman Begins(2005)
2.Batman(1989)	2.Batman Forever(1995)
3.Batman:Gotham Knight(2008)	3.The Dark Knight(2008)
4.The Batman/Superman Hour(1968)	4.The Batman(2004)
5.The Batman(2004)	5.Frankie and Johnny(1991)
6.The Dark Knight(2008)	6.The Exorcist(1973)

Table 4: A comparison between LSA and LDA techniques on the movie "The Matrix".

LSA	**LDA**
1.The Matrix (1999)	1.The Matrix (1999)
2.Computer Warriors (1990)	2.The Matrix Reloaded (2003)
3.Electric Dreams (1984)	3.Simulacrum (2009)
4.Willkommen in Babylon (1995)	4.Virus X (2010)
5.TRON 2.0 (2003)	5.Fallen Moon (2011)
6.Hackers (1995)	6.The Matrix Revolutions (2003)

4.3 Topic Model Comparison

We have performed several tests in order to evaluate which of the Topic Models was the best in defining the recommended movie list.

As described in section 3, the similarity of plots can be combined with the similarity of other features. We decided not to consider the features for this evaluation, so the LSA and LDA are compared considering only plots and none of the movie features. This decision was gained after a manual evaluation of the results of LDA and LSA with or without features. The manual evaluation showed several problems that have led us to formulate some considerations: (1) for each movie we have a broad list of actors, as IMDb reports the complete list including the background actors. Evaluating similarity starting from this list is really complicated and might lead to meaningless results; (2) in order to decrease computation cost, the features have been applied as a filter after the computation of the top 100 most similar movies according to the plot. Thus, the application of the features do not always improve the results gained by LSA or LDA.

To examine the quality of results of LDA and LSA, we chose three movies: two movies of a saga and a movie without a sequel. We calculated the five most similar movies for each target movie and analyzed the outcome. Table 3 shows the results for

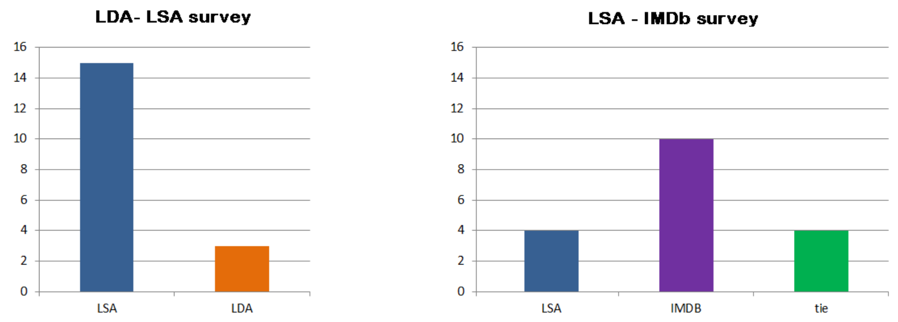


Figure 3: Performance of the topic models and IMDb on the two surveys.

the target movie "Batman Begins". All movies recommended by LSA belong to the series of Batman. The LDA list contains many movies of the saga, however the movies in fifth and sixth position have nothing in common with the target movie, these movies have a poor and short plot with words that are present also in the Batman Begins's plot (such as evil, sinister forces, family, prison). For the second movie "The Matrix" (see Table 4), LSA selected movies referring to the topics of computer, network, programmer, hacker. 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 as it can be seen in Table 5. For this kind of movies 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.

Table 5: A comparison between LSA and LDA techniques on the movie "Braveheart".

LSA	LDA
1.Braveheart (1995)	1.Braveheart (1995)
2.The Enemy Within (2010)	2.Windwalker (1981)
3.Journey of a Story (2011)	3.Lipgloss Explosion(2001)
4.Audition (2007)	4.Race for Glory (1989)
5.The Process (2011)	5.Voyager from the Unknown (1982)
6.Comedy Central Roast of William Shatner (2006)	6.Elmo Saves Christmas (1996)

4.4 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 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.

4.4.1 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" (see the survey displayed in left part of Figure 5). We collected 594 evaluations from 20 users in total. The evaluation results are shown in Table 6.

Table 6: A user evaluation of the two Topic Models.

judgement	LSA	LDA
"similar"	201	61
"not similar"	276	410
"I do not know"	120	129
movies without judgement	303	300

We also evaluated the behavior of the Topic Mod-

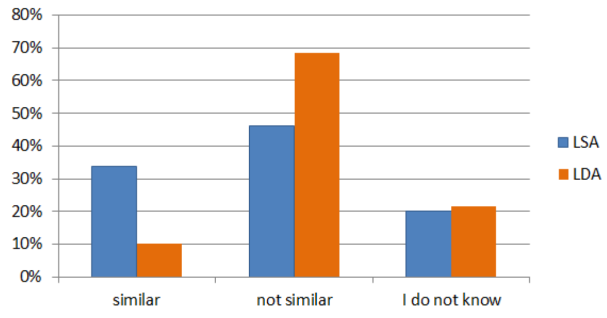


Figure 4: Percentage of users' judgements on LSA-LDA survey.

els 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 Figure 3). As expected from the previous comparison of the two models (reported in Tables 3,4,5), LSA supplied better recommendations than LDA. In Table 6 we have reported the total number of user judgements received (here we do not consider the movies for which users have not expressed a judgement).

4.4.2 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 the top 250 movies of IMDb[17]). 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 (see the survey displayed in right part of Figure 5). 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 Figure 3).

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

4.4.3 User Preference Evaluation

We added an in-deep evaluation of the users preferences for the 18 popular movies used in 4.4.2. This evaluation has been based on the precision measure computes by using the classification of recommendation results introduced in (Gunawardana and Shani, 2009) (see table 7) as

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

Table 7: Classification of the possible results of a recommendation of an item to a user (Gunawardana and Shani, 2009).

	Recommended	Not recommended
Preferred	True-Positive (*tp*)	False-Negative (*fn*)
Not preferred	False-Positive (*fp*)	True-Negative (*tn*)

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 8). 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.

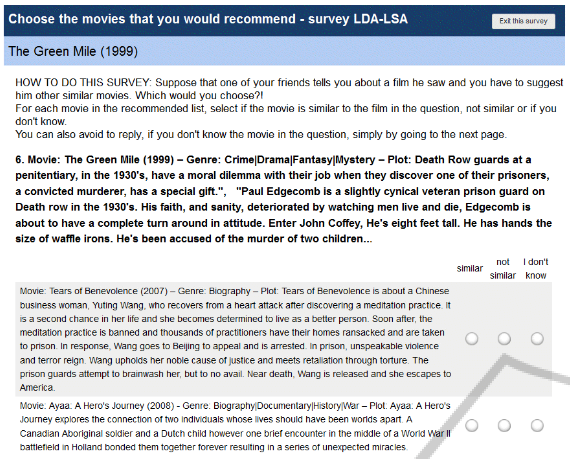
Table 8: Precision of the systems based on a punctual user preference evaluation.

	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

4.5 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: it is not able to suggest movies



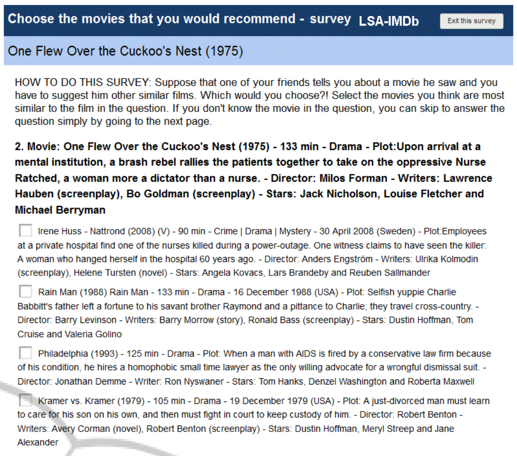
Choose the movies that you would recommend - survey LDA-LSA

The Green Mile (1999)

HOW TO DO THIS SURVEY: Suppose that one of your friends tells you about a film he saw and you have to suggest him other similar movies. Which would you choose?!
For each movie in the recommended list, select if the movie is similar to the film in the question, not similar or if you don't know.
You can also avoid to reply, if you don't know the movie in the question, simply by going to the next page.

6. Movie: The Green Mile (1999) – Genre: Crime|Drama|Fantasy|Mystery – Plot: Death Row guards at a penitentiary, in the 1930's, have a moral dilemma with their job when they discover one of their prisoners, a convicted murderer, has a special gift.", "Paul Edgecomb is a slightly cynical veteran prison guard on Death row in the 1930's. His faith, and sanity, deteriorated by watching men live and die, Edgecomb is about to have a complete turn around in attitude. Enter John Coffey, He's eight feet tall. He has hands the size of waffle irons. He's been accused of the murder of two children...

	similar	not similar	I don't know
Movie: Tears of Benevolence (2007) – Genre: Biography – Plot: Tears of Benevolence is about a Chinese business woman, Yuting Wang, who recovers from a heart attack after discovering a meditation practice. It is a second chance in her life and she becomes determined to live as a better person. Soon after, the meditation practice is banned and thousands of practitioners have their homes ransacked and are taken to prison. In response, Wang goes to Beijing to appeal and is arrested. In prison, unspeakable violence and terror reign. Wang upholds her noble cause of justice and meets retaliation through torture. The prison guards attempt to brainwash her, but to no avail. Near death, Wang is released and she escapes to America.	○	○	○
Movie: Ayaa: A Hero's Journey (2008) - Genre: Biography|Documentary|History|War – Plot: Ayaa: A Hero's Journey explores the connection of two individuals whose lives should have been worlds apart. A Canadian Aboriginal soldier and a Dutch child however one brief encounter in the middle of a World War II battlefield in Holland bonded them together forever resulting in a series of unexpected miracles.	○	○	○



Choose the movies that you would recommend - survey LSA-IMDb

One Flew Over the Cuckoo's Nest (1975)

HOW TO DO THIS SURVEY: Suppose that one of your friends tells you about a movie he saw and you have to suggest him other similar films. Which would you choose?! Select the movies you think are most similar to the film in the question. If you don't know the movie in the question, you can skip to answer the question simply by going to the next page.

2. Movie: One Flew Over the Cuckoo's Nest (1975) - 133 min - Drama - Plot:Upon arrival at a mental institution, a brash rebel rallies the patients together to take on the oppressive Nurse Ratched, a woman more a dictator than a nurse. - Director: Milos Forman - Writers: Lawrence Hauben (screenplay), Bo Goldman (screenplay) - Stars: Jack Nicholson, Louise Fletcher and Michael Berryman

☐ Irene Huss - Nattrond (2008) (V) - 90 min - Crime | Drama | Mystery - 30 April 2008 (Sweden) - Plot:Employees at a private hospital find one of the nurses killed during a power-outage. One witness claims to have seen the killer: A woman who hanged herself in the hospital 60 years ago. - Director: Anders Engström - Writers: Ulrika Kolmodin (screenplay), Helene Tursten (novel) - Stars: Angela Kovacs, Lars Brandeby and Reuben Sallmander

☐ Rain Man (1988) Rain Man - 133 min - Drama - 16 December 1988 (USA) - Plot: Selfish yuppie Charlie Babbitt's father left a fortune to his savant brother Raymond and a pittance to Charlie; they travel cross-country. - Director: Barry Levinson - Writers: Barry Morrow (story), Ronald Bass (screenplay) - Stars: Dustin Hoffman, Tom Cruise and Valeria Golino

☐ Philadelphia (1993) - 125 min - Drama - Plot: When a man with AIDS is fired by a conservative law firm because of his condition, he hires a homophobic small time lawyer as the only willing advocate for a wrongful dismissal suit. - Director: Jonathan Demme - Writer: Ron Nyswaner - Stars: Tom Hanks, Denzel Washington and Roberta Maxwell

☐ Kramer vs. Kramer (1979) - 105 min - Drama - 19 December 1979 (USA) - Plot: A just-divorced man must learn to care for his son on his own, and then must fight in court to keep custody of him. - Director: Robert Benton - Writers: Avery Corman (novel), Robert Benton (screenplay) - Stars: Dustin Hoffman, Meryl Streep and Jane Alexander

Figure 5: A screenshot of the beginning of a page of the surveys: LDA-LSA survey on the left, and LSA-IMDb survey on the right.

of the same saga and it suggests erroneous entries for movies that have short plot with words that are present also in the plot of the target movie (Sect.4.3), also the user evaluation underlines poor quality of the LDA recommendations (Sect.4.4.1,4.4.3);

- LSA achieves good performance on movie recommendations: it is able to suggest movies of the same saga and also unknown movies related to the target one (Sect.4.3), also the user evaluation underlines the good quality of the LSA recommendations (Sect.4.4.1,4.4.2,4.4.3);
- Although our system did not outperform the IMDb performance (Sect.4.4.2), an in-deep evaluation of users preferences has shown that the average precision gained by LSA is very close to the precision of IMDb (Sect.4.4.3); it would therefore be necessary to conduct an online analysis of the behaviour of the system in order to better understand how it performs compared to IMDb.

Finally, 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.Thus, our system can be also used to make recommendations when knowledge of users preferences is not available.

5 RELATED WORK

Recommendation algorithms are usually classified in content-based and collaborative filtering (Ekstrand et al., 2011). Collaborative filtering systems are widely industrially utilized, for example by Amazon, MovieLens and Netflix, 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, or combined different approaches as in hybrid systems.

Among recommendation systems (Adomavicius and Tuzhilin, 2005), content-based recommendation systems rely on item descriptions that usually consist of punctual data.

Jinni[18] 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 (Lee and Welsh, 2005) 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 (Koren et al., 2009). For example, in (Gemulla et al., 2011) 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 (Řehůřek and Sojka, 2010).

Also the LDA Topic Model has been already ap-

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

plied in recommendation systems to analyze textual information. In particular in (Jin et al., 2005) 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 (Krestel et al., 2009) 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 (Shi et al., 2013) 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 (Moshfeghi et al., 2011) 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 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, however it does not outperform the commercial approach (IMDb). However, it is important to notice that our system does not rely on human effort and can be ported to any domain where natural language descriptions exist. Moreover, a nice feature of the system is its independence from the movie ratings expressed by users; thanks to this independence, it is able to propose famous and beloved movies as well as old or unheard movies/programs that are similar to the content of the video the user has watched. This allows the system to find strongly related movies that other recommendation systems, such as IMDb, do not consider, as they has a low number of ratings.

The results shown in this paper highlight some limitations and stimulate some future directions for our research.

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 suggestion similar movies. Thus, we should couple our recommendation system with other techniques that do not totally rely on the plot.

Another major limitation is the description, i.e. the content of the plot. Most of the entries in the DBpedia collection do not have a plot. Even within a database like IMDb (the most accurate) not all the plots are described in a similar manner. Some are described by only one or two sentences, others, instead, are meticulously detailed. In addition to this, in most of the movies, the plot is not completely revealed, and this leads to have several movie descriptions that are partial.

While LDA deals with polysemy issue, LSA does not. This problem can be faced by making use of a lexical database as WordNet[19]. 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 (Navigli, 2009). The semantic relationships between synsets can be used for enhancing the keyword meaning by adding all its hypernyms and hyponyms (Po and Sorrentino, 2011; Sorrentino et al., 2010).

We used our system to compute the similarity of a movie with other movies in the database. However, in general we can use it to evaluate the similarity of textual descriptions such as plots of movies not present in the DB or news, book plots, book reviews etc.

For example, the system can find movies that contain a story similar to the one tell 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).

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

ACKNOWLEDGEMENTS

We want to express our gratitude to Tania Farinella, Matteo Abbruzzo and Olga Kryukova, master students in Computer Engineering and Science at the 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 Thomas Werner and Andreas Lahr[20], founders of vfree.it, for their suggestions and valuable comments on the paper.

REFERENCES

Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Debnath, S., Ganguly, N., and Mitra, P. (2008). 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. ACM.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230.

Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Found. Trends Hum.-Comput. Interact.*, 4(2):81–173.

Farinella, T., Bergamaschi, S., and Po, L. (2012). A non-intrusive movie recommendation system. In *OTM Conferences (2)*, pages 736–751.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). 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. ACM.

Griffiths, T., Steyvers, M., and Tenenbaum, J. (2007). Topics in semantic representation. *Psychological Review*, 114(2):211–244.

Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research*, 10:2935–2962.

Jin, X., Mobasher, B., and Zhou, Y. (2005). A web recommendation system based on maximum entropy. In *ITCC (1)*, pages 213–218. IEEE Computer Society.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Krestel, R., Fankhauser, P., and Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In Bergman, L. D., Tuzhilin, A., Burke, R. D., Felfernig, A., and Schmidt-Thieme, L., editors, *RecSys*, pages 61–68. ACM.

Lee, M. D. and Welsh, M. (2005). An empirical evaluation of models of text document similarity. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, CogSci2005, pages 1254–1259. Erlbaum.

Moshfeghi, Y., Piwowarski, B., and Jose, J. M. (2011). 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 '11, pages 625–634, New York, NY, USA. ACM.

Musto, C. (2010). Enhanced vector space models for content-based recommender systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 361–364, New York, NY, USA. ACM.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2).

Park, L. A. F. and Ramamohanarao, K. (2009). An analysis of latent semantic term self-correlation. *ACM Trans. Inf. Syst.*, 27(2):8:1–8:35.

Po, L. and Sorrentino, S. (2011). Automatic generation of probabilistic relationships for improving schema matching. *Inf. Syst.*, 36(2):192–208.

Rashid, A. M., Karypis, G., and Riedl, J. (2008). Learning preferences of new users in recommender systems: an information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100.

Řehůřek, R. and Sojka, P. (2010). 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. ELRA. http://is.muni.cz/publication/884893/en.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18:613–620.

Shi, Y., Larson, M., and Hanjalic, A. (2013). Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Trans. Intell. Syst. Technol.*, 4(1):16:1–16:19.

Sorrentino, S., Bergamaschi, S., Gawinecki, M., and Po, L. (2010). Schema label normalization for improving schema matching. *Data Knowl. Eng.*, 69(12):1254–1273.

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