Learning from User-generated Data SS2016 – Task 2

Group D

Jurgen Ratzenbeck
Mat.nr.: 1256030
juergen.ratzenboeck@gmail.com
Score Percentage: 25%

Jonathan-Edwin Asamoah
Mat.nr.: 1457554
asamoahjonathan@outlook.com
Score Percentage: 25%

Mhd Mousa Hamad
Mat.nr.: 1556686
mhd.mousa.hamad@gmail.com
Score Percentage: 25%

Luka Vukas
Mat.nr.: 1557457
vukaslukaa@gmail.com
Score Percentage: 25%

1. DESCRIPTION OF TASK 2

In this task, we should predict a set of movie ratings by adding the ratings to 200.000 (user, movie) tuples. To fulfill the task, three different datasets are provided. The first one contains 800.000 (user, movie, rating) triples which represent previous movie ratings. This file should be used in an item-based collaborative filtering approach to find the similarity between movies based on these ratings. The second file contains more information about the movies namely, title, release year and genres. This information should be used to crawl online information sources to collect more information about these movies. Then, the collected information is used to enhance the similarity results between movies, which in turns enhances the prediction accuracy. The third file contains a set of (user, movie) tuples which should finally be enriched by the predicted rating of that user for that movie.

The remainder of this paper is organized as follows. In section 2 we provide an overview and a short analysis on the provided datasets. Section 3 explains how we addressed this task and which algorithms and resources we used to compute the missing movie ratings. This section also proposes our approach to improve the results of IBCF by incorporating the information collected from external sources. Section 4 how we tested our approach using different configurations and the setups conducted for these configurations. Section 5 shows our results for different settings, and the last section wraps up the task by concluding it.

2. DATA ANALYSIS

The data analysis for task 2 is based on the information that was gathered in the data analysis for task 1. The results of the data analysis for task 1 involved the analyses of the training dataset, and the predict dataset. The analysis of the training dataset showed that the dataset contains 800167 (user, movie, rating) triples with the highest density at a rating of four. The analysis of the predict dataset showed that it contains 200042 (user, movie) tuples with 25 of them not part of the training dataset.

The new movies dataset contains 3883 (id, title, genres) triples that fit to the given data from task 1. It is used to gather additional information about the movies, three different data sources were used. Figure (????) shows that using OMDB, we were able to gather information for 3676 movies (94%) from the dataset, where using TMDB, 3575 movies (92%) were enriched and finally using DBpedia, 3214 movies (82%) were enriched by external information.

Based on this observation and the high density of the features, OMDB was used as the basic information source. Whereas, the other two sources were used to fill any missing information from OMDB, extend this information using other meaningful missing features and merge any available textual data to enrich it the description of the movies.

This process leads to one single dataset with 95 Features, containing the modified features form the OMDB enriched by the features from the two other sources. After removing all the features that had an extremely low density, was empty, or was highly correlated we end up with 19 features. Figure (1) shows the density for these 19 features which was used in combination with other subjective criteria to choose the final features that would be considered in our approach to predict the ratings.

3. PROPOSED METHOD

For this task, we implemented a Python program that crawls three different web sources (omdb, tmdb, dbpedia). The program uses the crawled data along with an implemented item-based collaborative filtering algorithm to compute the missing predictions. Figure (2) illustrates the proposed approach.

3.1 Item-Based Collaborative Filtering (IBCF)

This approach uses the similarity between items to recommend similar items for a user or predict how would a user rate an item. When users rate an item, that item's similar

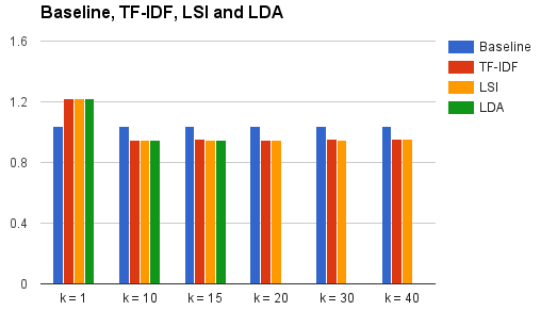

Figure 1: Density of Features

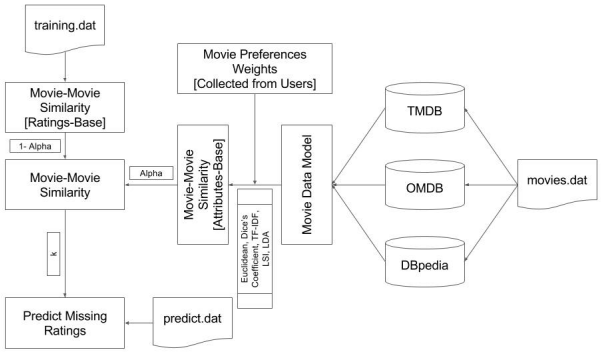

Figure 2: Proposed Approach

items are picked from the existing system model and added to the user's recommendations. This approach is usually preferred to other memory-based approaches (user-based) when the system has more users than items as it leads to more stable rating distribution.

Figure (2) shows the basic steps of IBCF approach to predict how a user would rate an item. First the model has to be built by finding the similarity between all pairs of items. Then, the most similar k items which are already rated by the user are selected to compute the predicted rating of that user of the item. This computation of the rating is simply the average of that user's ratings for the selected items.

To find the similarity between two items, one can use previous users' ratings of these items and/or items' attributes. In this task, previous users' ratings are provided along some other attributes for movies which are the items of this task. The provided attributes for movies were enough to search for them in the some public movie databases and other public resources and collect some other attributes for these movies. The similarity between two item is then calculated using a weighted sum scheme as in the following equation:

$$sim(x, y) = (1 - \alpha) \times sim_r(x, y) + (\alpha) \times sim_a(x, y) \quad (1)$$

Where:

sim_r is the similarity between two movies (m1, m2) based on previous ratings

sim_a is the similarity between two movies (m1, m2) based on the collected attributes

α is a weighting factor, in practice this factor performed best when set to (1.0), which means using only the attributes-based similarity

3.1.1 Ratings-Based Similarity

This similarity can take many forms, such the correlation between the compared ratings or the cosine of the angle between the vectors that are formed with these ratings. These similarity measures were explored in more details and experimented in the first task. The cosine similarity performed the best with IBCF. Therefore, it the only method that was experimented in this task.

3.1.2 Attributes-Based Similarity

In this task, we were provided by the title, year and genres of the movies. We used this information to crawl other information about these movies from different online open resources. The crawled information were aggregated and used to measure how movies are similar to each other.

Data Sources

OMDB and **TMDB** are open movie databases accessible via custom web APIs. They provide much information about movies like (genres, actors, directors, writers, ratings,). The provided ratings are either automatic ratings based on critics and reviews of the movies (Tomatometer from TMDB) or average user ratings collected from different sources (IMDB, TMDB).

DBpedia is a data-oriented interface to Wikipedia community. It combines the information spread across many Wikipedia pages into a single integrated database. This information is accessed using SPARQL which is a semantic query language for databases that store data Resource Description Framework (RDF) format. A lot of information about movies can be crawled from DBpedia using SPARQL.

Twitter is a popular social network which users can use to write about anything. It provides a Streaming and Search APIs to access users' posts (Tweets). The retrieved Tweets can be filtered using different keywords (Hashtags). These Tweets may contain users' reviews about movies which can be mined to extract the average ratings of movies using sentiment analysis.

OMDB, TMDB and DBpedia were crawled in this task to get more information about the movies, while Twitter was discarded as "the Twitter Search API searches against a sampling of recent Tweets published in the past 7 days" [According to Twitter's Search API documentation].

Similarity Measures

The crawled information was checked and only important attributes were kept to be processed by the similarity model. These attributes were categorized into seven categories:

- Year: contains the release year of the movie
- Rating: contains all crawled ratings for the movie (Tomatometer, IMDB Rating, Tomato User Rating,)
- Genres: contains the genres of the movie
- Stakeholders: contains all the main players in the movie production (writers, actors, directors and production companies)
- Description: contains all text descriptions crawled from the different sources
- Other: contains any other important features (awards,)

For each category, a specific similarity measure is used. The similarity between two movies is calculated as a weighted sum of the similarities between their categorized attributes.

The similarity between "Year" and "Rating" attributes is measured using *Euclidean Distance*. The year is a scalar value were the rating is a vector of values each of which represents different crawled rating.

The similarity between "Genres" and "Stakeholders" attributes is measured using *Dice's Coefficient*. Dice's Coefficient is a common similarity method between two sets. Genres and full stakeholders' names were considered the items of the sets. This ensures that two different stakeholders with the same first or last name won't have any similarity. The similarity between stakeholders is calculated as a weighted sum of the similarities between each set of stakeholders (writers, actors, directors and production companies) using the same similarity measure.

The similarity between "Other" attributes is measured based on the attribute. Currently, we are considering only the awards which is measured using Dice's Coefficient.

The similarity between "Description" attributes is measured using one of the corpus-based similarity measures namely, Term Frequency-Inverse Document Frequency (TF-IDF), Latent Semantic Indexing (LSI) and Latent Dirichlet allocation (LDA). We defined a virtual document for each movie which is all the crawled text descriptions of it. We then used all virtual documents of all movies as our corpus to build these models.

TF-IDF is a numerical statistic that reflects how important a term (word) is to a document in a corpus. It is used in Vector Space Model where a matrix containing these numerical statistics for a term in a document (rows represent each document and columns represent unique terms) is constructed from a corpus. Documents are then compared by taking the cosine of the angle between the two vectors formed by any two rows.

LSI assumes that terms that are close in meaning will occur in similar pieces of text. A matrix containing a numerical statistic for a term in a paragraph (rows represent unique terms and columns represent each paragraph) is constructed from a corpus. Singular Value Decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Terms are then compared by taking the cosine of the angle between the two vectors formed by any two rows.

LDA is an example of a topic model that allows terms or documents to be represented by a set of probabilities of falling into some unobserved groups (topics). It posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. Terms or documents are then compared by taking the cosine of the angle between the two vectors formed by any two sets of probabilities. LDA is similar to Probabilistic LSA (pLSA).

LSA and LDA uses a predefined number of topics which represent the number of latent factors that should be extracted and used to represent the terms and documents.

4. EXPERIMENTAL SETUP

As the final predictions of the task are evaluated using Root Mean Square Error (RMSE), we used this measure to evaluate our models. To make sure that the evaluation is not biased, we used 10-fold cross validation on the provided "training.dat" file as the true ratings are provided. Using cross-validation will ensure that no triples (user, item, rating) are used in both training (building) and evaluating the model.

Tuning all the weights automatically is not feasible as each single evaluation takes from (10-24) hours. We started by determining the weights of the attributes-based movie similarity. We setup an online excel document and asked some friends who watch movies a lot to order the preferences of selecting a movie to watch from the most important (setting its order to one) to the least important (setting its order to 9). These preferences are the attributes that we are considering in measuring the similarity between movies. Each user was given a choice to provide his name or keep it anonymous. People's preferences are kept visible to everybody. We then averaged the order of each attribute and extracted the weight of it by subtracting this average from 9. Table (1) shows the preferences and their average and calculated weight.

As there is no clear documented way for selecting the number of topics (latent factors) for LSI and LDA, this parameter should be tuned automatically. Unfortunately, this was not feasible for this task due to the lack of resources. Professor Wray Buntine, Monsh University, wrote as an answer to online question[1] about how to set this parameter: "With 20,000 documents using a good implementation of HDP-LDA with a Gibbs sampler I can sometimes get K2000",

[1]https://www.quora.com/Latent-Dirichlet-Allocation-LDA-What-is-\
\the-best-way-to-determine-k-number-of-topics-in-topic-modeling

Attribute	Average	Weight
Year	4.22	5
Rating	3.00	6
Genres	2.89	7
Actors	4.00	5
Production Companies	6.78	3
Directors	5.44	4
Writers	5.44	4
Description	3.11	6
Other	7.89	2

Table 1: User Preferences for Selecting Movies

	IBCF
k = 1	1.037
k = 10	1.115
k = 15	1.123
k = 20	1.129
k = 30	1.134
k = 40	1.135

Table 2: First Task Results for IBCF

"Now inspecting the 2000 topics, maybe 1600 have good quality comprehensibility". As we have 3883 virtual documents and these virtual documents are relatively short (compared to web-page discussing a topic), we set this parameter to (150).

Fixing these parameters, we first evaluated our approach using only the attributes-based similarity between movies $\alpha = 1.0$. The following configurations were used:

- **TF-IDF** with $k = 1, 10, 15, 20, 30, 40$
- **LSI** with $k = 1, 10, 15, 20, 30, 40$
- **LDA** with $k = 1, 10, 15, 20, 30, 40$

After determining the best model for corpus-based similarity (LSI) with the best k value ($k = 15$), we checked the results of the first task to determine which k performed the best. We compared results of each k (ratings-based and attribute-based) and fixed one value in between $k = 10$. Fixing k, we then evaluated our approach again using both similarities (ratings-based and attribute-based). The following configurations were used:

- $\alpha = 0.2, 0.4, 0.6, 0.8$

To compare the results of these different configurations, we introduced a simple mean-user-ratings approach as a baseline. This approach simply predict the missing rating of a user for an item by averaging all his previous ratings.

5. RESULTS

Table (2) shows the evaluation results from the first task for IBCF using the cosine similarity measure. We can see that the best performance was at $k = 1$

Table (3) and Figure (3) show the results of the first evaluation set along with the baseline. From them, we can see that all the tested text similarity mesures (TF-IDF, LSI, LDA) performed very closely with LSI having the best performance at $k = 15$.

	Baseline	TF-IDF	LSI	LDA
k = 1	1.036	1.224	1.222	1.221
k = 10	1.036	0.948	0.946	0.947
k = 15	1.036	0.952	**0.944**	0.947
k = 20	1.036	0.946	0.947	0.947
k = 30	1.036	0.951	0.949	0.947
k = 40	1.036	0.954	0.955	0.947

Table 3: First Evaluation Set - $\alpha = 1.0$

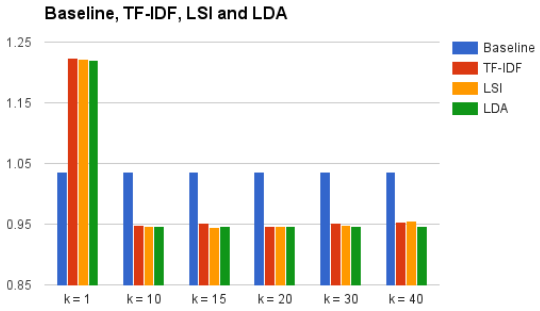

Figure 3: First Evaluation Set - $\alpha = 1.0$

Table (4) shows the results of the second evaluation set. In this set, we fixed $k = 10$ and run using only LSI model as it has the best performance in the first evaluation set. $k = 10$ is fixed between the best values of k in IBCF in the first and second task. It is closer to the best k in the second task as it's performance is relatively better.

6. CONCLUSIONS

TF-IDF, LSI and LDA were so close in their performance having LSI as the best performer in combination with $k = 10$. The evaluated text models didn't affect much the final results as there are much other attributes affecting the similarity between two movies. Tuning all parameters for these attributes needs much more resources than we have. Although the results were so close, we preferred to go with best performer to predict the final missing ratings in the file "predict.dat"

	LSI
alpha = 0.0	1.115
alpha = 0.2	0.947
alpha = 0.4	0.948
alpha = 0.6	0.949
alpha = 0.8	0.949
alpha = 1.0	0.946

Table 4: Second Evaluation Set - $k = 10$