Learning from User-generated Data SS2016 – Task 3

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 3

In this task, we should predict a set of movie ratings for 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. The second contains more details about the movies namely, title, release year and genres. The third contains some information about the users who rated the movies namely, gender, age range, job and zip-code. Information provided by these three files combined with extended information crawled from multiple on-line movie sources should be used in experimenting and evaluating different hybrid recommendation systems. The fourth file contains a set of (user, movie) tuples which should finally be enriched by predicted user ratings for the movies using the best evaluated recommendation system.

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. It also proposes our parallelized and monolithic approaches to predict the missing ratings. Section 4 presents how we tested our approaches using different configurations and also the setups conducted for these configurations. Section 5 shows the results for different setups, and the last section wraps up the task by concluding it.

2. DATA ANALYSIS

Data analysis was performed mainly in tasks 1 and 2. The first file of this task contains 800167 (user_id, movie_id, rating) tuples used for training with the highest rating density at a four. The third file of this task contains 200042 (user_id, movie_id) tuples used for predicting the final ratings with 25

Figure 1: Percentage of returned movie information

Figure 2: Histograms for users set

new movie_ids which don't exist in the training set.

The second file contains 3883 (id, title, genres) triples which fit the given rated (to be rated) movies for this task. Three different on-line movie sources were crawled to gather additional information about the movies (OMDB, TMDB, DBPedia). Figure (1) shows the percentage of movies that we were able to enrich from each of these sources with (94%, 92%, 82%) from (OMDB, TMDB, DBPedia) respectively.

OMDB was used as the main information source, where TMDB and DBPedia were used to enrich this source either by filling in the missing values or by extending the basic information. This process led to 95 features for each movie, which were analyzed and processed to end up with 17 important features that have relatively high densities.

The third file contains 6040 (id, gender, age, job, zip-code) quintuples which also fit the users who has (should have) ratings in the datasets of this task. The user information provided was complete and sound (no missing or unknown values). Figure (???) shows histograms for this set. The users tend to be males (71%) and in the age range of [25, 34] (34%).

3. PROPOSED METHOD

For this task, we worked on two sets of approaches: early-fusion (parallelized design) and late-fusion (monolithic design). For the early-fusion approach, we boosted item-based collaborative filtering using content-based movies similarity, and user-based collaborative filtering using content-based users similarity. [???] Finally, we boosted matrix factorization approach using ??????????. For the late fusion approach, we combined the results of multiple recommendation systems using a regression model.

3.1 Early-Fusion Approach

This approach adapts default algorithms to deal with mod-

Figure 3: Item-based collaborative filtering approach

Figure 4: User-based collaborative filtering approach

ified hybrid input data. This data contains combinations and/or augmentations of data from multiple different sources.

3.1.1 Item-based Collaborative Filtering (IBCF)

This approach measures the similarity between items to recommend similar items for a user or predict how would he/she rate an item. When users rate an item, its similar items are picked from the existing system model and added to user's recommendations. This approach is usually preferred over other memory-based approaches (user-based) when the system has more users than items as it leads to a more stable rating distribution.

Figure (3) shows the basic steps of IBCF approach to predict how a user would rate an item. First a model is built by finding the similarity between all pairs of items. Then, the most similar k items which have been already rated by the user are selected to compute the predicted rating of that user for the item. This computed rating is the average rating of all selected items rated by that user.

The similarity between two items is calculated using a weighted sum scheme that combines the ratings-based similarity that uses previous users' ratings and the content-based similarity that uses the crawled information from on-line movie sources. The following equation shows how the similarity between two movies is calculated:

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

Where:

sim_r is the ratings-based similarity between the movies (m1, m2)

sim_a is the content-based similarity between the movies (m1, m2)

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

The crawled movie attributes were divided into seven categories as follows:

- Year: contains the release year of the movie

- Rating: contains all crawled ratings of the movie (Tomato Meter, IMDB Rating, Tomato User Rating,)

- Genres: contains genres of the movie

- Stakeholders: contains all 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. Then, the similarity between two movies is calculated as a weighted sum of the similarities between their categorized attributes. The measures for each attribute category were discussed in more details in task 2.

3.1.2 User-based Collaborative Filtering (UBCF)

This approach measures the similarity between users to recommend new items for a user or predict how would he/she rate an item based on the ratings of other similar users. This approach is widely used and has good recommendation results but it does not scale with the number of users when it goes very high.

Figure (4) shows the basic steps of UBCF approach to predict how a user would rate an item. First a model is built by finding the similarity between all pairs of current users. Then, the most similar k users which have already rated the item are selected to compute the predicted rating of that user for the item. This computed rating is the average rating of all selected users.

The similarity between two users is calculated using a weighted sum scheme similar to the one used for calculating the similarity between two movies. This scheme also combines ratings-based similarity and content-based similarity that uses the users information provided in the third file (the users set).

Content-based similarity between two users is calculated as a weighted sum of the individual similarities between their attributes. Age and zip-code similarities were measured using the Euclidean distance. Gender and job similarities were measured using a binary scheme, where two values are either similar (when they match exactly) or not similar (when they differ).

The weights for combining individual similarities into the final content-based similarity between two users were set manually due to the lack of evaluation time and resources.

3.1.3 Matrix Factorization????

3.2 Late-Fusion Approach

This approach aggregates the outputs of multiple recommendation systems into one final recommendation. The aggregation could be using weighting or voting schemes. In the weighting scheme, the results of all individual recommendation systems are combined, so the final result may differ from all individual results. In the voting scheme, each individual recommendation system votes for a result (for example, increment its occurrences by one); the final recommendation is one of the individual recommendations and it is the one with the highest votes.

Figure (5) shows the basic steps of late-fusion approach. Five different recommendation systems were aggregated using a weighting scheme. To set the weights, we used a K-Nearest-Neighbor regression model. The outputs of these five recommendation systems are used as inputs for the regression model, and the final output is the final submitted rating.

The five recommendation systems that were aggregated are:

- Item-based collaborative filtering

- User-based collaborative filtering

- Matrix factorization

Figure 5: Late-fusion approach

Attribute	Weight
Year	5
Rating	6
Genres	7
Actors	5
Production Companies	3
Directors	4
Writers	4
Description	6
Other	2

Table 1: Weights of movie feature

- Content-based collaborative filtering (with movies information as the content)

- Content-based collaborative filtering (with users information as the content)

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 for content-based similarity measures (for movies and users) is not feasible as each single evaluation takes from (10-40) hours. We set these weights manually using a simple voting approach. Multiple persons were asked to rank their preferences and the average rankings were used to compute the weights (more details are available in task 2). Tables (1, 2) show the weights for each set of attributes for movies and users respectively.

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", "Now inspecting the 2000 topics, maybe 1600 have good quality comprehensibility". As we have 3883 virtual documents and these virtual documents are relatively short (com-

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

Attribute	Weight
Gender	5
Age	4
Job	3
Zip-code	2

Table 2: Weights of movie feature

	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 3: First Task Results for IBCF

	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 4: First Evaluation Set - $\alpha = 1.0$

pared 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 predicts the missing rating of a user for an item by averaging all his previous ratings.

5. RESULTS

Table (3) 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 (4) and Figure (6) 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 $k = 15$.

Table (5) 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.
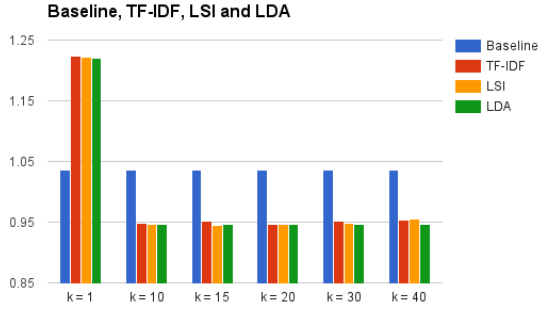
Figure 6: First Evaluation Set - $\alpha = 1.0$

	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 5: Second Evaluation Set - $k = 10$

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 the final results much as there are 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 close, we preferred to go with best performer to predict the final missing ratings in the file "predict.dat"