Profile based retrieval: Project Report

Profile-based movies recommender

Federico Rodigari: federico.rodigari@alumnos.upm.es

Department of Computer Science
Master EIT Digital in Data Science

April 8, 2021

Contents

1 Scope

The last decade has opened in the age of big data and data economy, in which computational power and memory combined make it possible to extract from data important knowledge, insights, and potential. This results in significant challenges and opportunities that are challenging the domains of research, healthcare and business. Both public and private sector industries generate, store, and analyze big data with an aim to improve the services they provide.
In particular, very well known platforms such as Netflix, Spotify and Amazon are asked to address the needs of the customers by providing the users with recommendations based on previous interactions and preferences. However, this can not be manually since the multitude of users and need a well-consolidated system. We refer to those systems with the name of "Recommendation Systems". Recommendation Systems provide a solution for the information flood each user has to face every day during their interaction with online platforms such as the one mentioned before. Aiming to provide users with the ability to find products that may be of their interest, make up the target of these systems in an automatic, fast and efficient way. Their recommendations are based on the analysis of previous user behavior, with respect to the evaluation of products as well as the recognition of the similarity between different users and products. This occurs, in order to derive a prediction of which products a user will consider interesting. There are mainly two types of recommendation systems: Collaborative and Content-based filtering. In this project we will follow the second method, which uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback. The scope of this project is, at the end, to merge knowledge of Natural Language Process, statistics and programming to come up with a systems that correctly recommends users with movies based on their profile. The project is carried out in Python language with the help of Pandas, Scikit-learn, NLTK and Numpy and the code is available in the following Github repository.

2 Data understanding

The first stage of the development of a recommendation system relates to the clear understanding of the data provided. In this section we will go trough the whole dataset and outline a pathway to maximize the potential of data to solve our data science problem. The dataset chosen for the purpose of the project describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 27753444 ratings and 1108997 tag applications across 58098 movies. These data were created by 283228 users between January 09, 1995 and September 26, 2018.

- **ratings.csv**: contains all ratings, each line of this file represents one rating of one movie by one user in the format userId, movieId, rating, timestamp.
- **tags.csv**: contains all tags, each line of this file represents one tag applied to one movie by one user, and has the following format: userId, movieId, tag, timestamp.
- **movies.csv:** contains all movie information, each line of this file represents one movie, and has the following format: movieId , title, genres.
- **links.csv**: contains all identifiers that can be used to link to other sources of movie data,each line of this file after the header row represents one movie, and has the following format: movieId, imdbId, tmdbId.
- **genome-scores.csv**: contains movie-tag relevance data in the following format: movieId ,tagId, relevance.
- **genome-tags.csv**: provides the tag descriptions for the tag IDs in the genome file, in the following format: tagId, tag.

The tag genome is a data structure that contains tag relevance scores for movies. The structure is a dense matrix: each movie in the genome has a value for every tag in the genome. The tag genome encodes how strongly movies exhibit particular properties represented by tags (atmospheric, thought-provoking, realistic, etc.). The tag genome was computed using a machine learning algorithm on user-contributed content including tags, ratings, and textual reviews. However the main data used for the assignment will be genomes, movies and ratings.

3 Preprocessing data

In this section we explain how we conduct a variable transformation, selection and preprocessing that we use for the development of the system and results in the final dataset that will be used for the recommendation process.

3.1 Data cleaning

In the first step of this phase, after importing all the data, the two dataset related with the genome were merged in order to produce a table that shows the tags in text with the Ids, the relevance and the movie associated. The first cleaning operation is to remove all the tags that have *relevance lower than 0.3* since they would not be useful for the purposes. After this operation, the tags available decreases from 14862528 to 1471046.

3.2 Data preparation

At this point, the focus of the work is on the remaining tags. After a first overview of the tags provided, I noticed that some tags were similar and only differentiated between each other by plural or gender form, here an example:

	movieId	tagId	relevance	tag
14840190	4472	1127	0.39900	zombie
14840198	4480	1127	0.36600	zombie
14840211	4493	1127	0.31600	zombie
14840215	4497	1127	0.92000	zombie
14840250	4532	1127	0.96500	zombie
...	...	...	...	...
14862143	160112	1128	0.75275	zombies
14862179	162082	1128	0.98425	zombies
14862222	164367	1128	0.97300	zombies
14862316	168498	1128	0.88225	zombies
14862368	170945	1128	0.33975	zombies

Figure 1: Raw dataframe

For this reason, thanks to the package NLTK, the tags were stemmed by using the function **PorterStemmer().** PorterStemmer uses *"Suffix Stripping"* to produce stems, the algorithm does not follow linguistics rather a set of rules for different cases that are applied step by step to generate stems. This is the reason why PorterStemmer does not often generate stems that are actual English words. It does not keep a lookup table for actual stems of the word but applies algorithmic rules to generate stems by deciding whether it is wise to strip a suffix. After this operation, the dataframe is as follow:

	movieId	tagId	relevance	tag	stemmed
9	10	1	0.99950	007	007
366	380	1	0.58500	007	007
497	517	1	0.32275	007	007
607	648	1	0.62200	007	007
682	743	1	0.54100	007	007
...	...	...	...	...	...
14862143	160112	1128	0.75275	zombies	zombi
14862179	162082	1128	0.98425	zombies	zombi
14862222	164367	1128	0.97300	zombies	zombi
14862316	168498	1128	0.88225	zombies	zombi
14862368	170945	1128	0.33975	zombies	zombi

Figure 2: Stemmed tags dataframe

Once the tags are stemmed, the dataframe with the tags and the one with the scores are merged to have a better understanding on the relation between the tags and the movies. Now the dataframe is grouped by the stemmed tags in text and the Ids are reassigned according to them. This operation allow to get rid of the differentiation between similar tags and its relevance. This can be noticed in the following table:

	movieId	tagId	relevance	tag	stemmed	grpId
14840190	4472	1127	0.39900	zombie	zombi	1078
14840198	4480	1127	0.36600	zombie	zombi	1078
14840211	4493	1127	0.31600	zombie	zombi	1078
14840215	4497	1127	0.92000	zombie	zombi	1078
14840250	4532	1127	0.96500	zombie	zombi	1078
...	...	...	...	...	...	...
14862143	160112	1128	0.75275	zombies	zombi	1078
14862179	162082	1128	0.98425	zombies	zombi	1078
14862222	164367	1128	0.97300	zombies	zombi	1078
14862316	168498	1128	0.88225	zombies	zombi	1078
14862368	170945	1128	0.33975	zombies	zombi	1078

Figure 3: Reindexed dataframe

The next operation to be done for our recommendation system is to create a new column in the dataframe with all the tags related to a specific film. This will allow us to perform all the similarity measures needed for the development.

At the end of this process some statistics are produced to achieve the final dataframe that will permit the development of the system. In this way, for each movie a the number of ratings, the rating mean and the rating median is performed obtaining the final implementation on which the similarities will be performed.

```
df_data_with_tags
```

	movieId	title	genres	rating_mean	rating_median	num_ratings	movie_tags
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	3.886649	4.0	68469.0	1035 992 664 412 338 468 355 504 230 246 675 7...
1	2	Jumanji (1995)	Adventure|Children|Fantasy	3.246583	3.0	27143.0	690 468 355 230 742 663 454 21 415 920 389 48 ...
2	3	Grumpier Old Men (1995)	Comedy|Romance	3.173981	3.0	15585.0	387 926 911 299 902 397 412 469 468 654 1070 9...
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	2.874540	3.0	2989.0	387 640 1035 107 299 911 614 412 425 468 201 1...
4	5	Father of the Bride Part II (1995)	Comedy	3.077291	3.0	15474.0	387 926 1035 299 902 922 469 468 348 1070 414 ...
...	...	...	...	...	...	...	...
13171	185435	Game Over, Man! (2018)	Action|Comedy	2.562500	3.0	64.0	8 926 468 540 770 390 1123 726 230 545 742 21 ...
13172	185585	Pacific Rim: Uprising (2018)	Action|Fantasy|Sci-Fi	2.830128	3.0	312.0	107 890 933 1092 468 251 46 1070 167 889 919 1...
13173	186587	Rampage (2018)	Action|Adventure|Sci-Fi	2.945755	3.0	212.0	832 1035 992 690 664 412 308 338 468 478 872 3...
13174	187593	Deadpool 2 (2018)	Action|Comedy|Sci-Fi	3.838947	4.0	1633.0	926 181 251 645 390 454 749 1000 415 1096 462 ...
13175	187595	Solo: A Star Wars Story (2018)	Action|Adventure|Children|Sci-Fi	3.464356	3.5	505.0	832 992 690 412 468 251 58 390 355 742 364 353...

13176 rows × 7 columns

Figure 4: Final dataframe

4 Data encoding

In this stage the data will be encoded. There are different way to perform this task but the most adequate for this case is to produce a TF-IDF matrix and perform Cosine Similarities on the values of the sparse matrix generated. The result of this process is the kernel of the recommendation system. From this point, with analogues techniques, the user profile is constructed and is ready to be recommended.

4.1 TF-IDF matrix

Term Frequency-Inverse Document Frequency(TF-IDF) is used in Information Retrieval for feature extraction purposes and it is a sub-area of Natural Language Processing. TF-IDF is a measure used to evaluate how important a word is to a document in a document corpus. The importance of the word increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

In this project, the focus of TF-IDF encoding is to weigh a term that in our case is a tag in text form according to the importance of the term within the document, this means that words such as the one available in a stop-word dictionary which are likely to show up in every movie description but are not useful for the user-recommendation, will be weighed less than words that are more unique to the content that we are recommending. For this reason, in the preparation step mentioned in 3, we did not take into account the process of removing stop-words.

This operation is performed tanks to the package *Scikit-learn* that provides the function *TfidfVectorizer*.

4.2 Cosine Similarity

Cosine similarity is a metric used to determine how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. In this context, the two vectors I am talking about are arrays containing the word counts of two documents. When plotted on a multi-dimensional space, where each dimension corresponds to a word in the document, the cosine similarity captures the the angle difference of the documents and not the magnitude as it would be for the Euclidean distance. The advantage of the Cosine similarity compared to the Euclidean distance is that given two documents that has a long distance in between, the angle gives a more accurate measure of similarity. That means that even distant terms could result similar if treated with the Cosine similarity.

In the case of the recommendation system, Recommending content involves making a prediction. The prediction is mainly related to the behaviour of the customer. This means that, in the case of a movie recommendation system, the prediction is based the about how likely it is that a user is going to like the recommended content. This is done at the end by the generation of a matrix that performs the similarities between the movies and score them accordingly.

In this case, the words taken into account for the matrix are the tags that were stemmed in the previous section, a sample of the matrix is provided as follows:

	1	2	3	4	5	6	7	8	9	10	...
1	1.000000	0.462745	0.215649	0.220860	0.228145	0.410830	0.296765	0.269200	0.076516	0.241485	...
2	0.462745	1.000000	0.183530	0.235638	0.171096	0.143772	0.288950	0.290667	0.278500	0.288449	...
3	0.215649	0.183530	1.000000	0.263559	0.503608	0.167767	0.357936	0.154670	0.070063	0.191633	...
4	0.220860	0.235638	0.263559	1.000000	0.287089	0.124760	0.401416	0.356245	0.077829	0.099731	...
5	0.228145	0.171096	0.503608	0.287089	1.000000	0.126927	0.407728	0.175734	0.083730	0.179651	...

Figure 5: TF-IDF matrix

4.3 User profile

The next step of the development of the application is to construct a user profile based on a methodology. The approach persecuted in this step is to build the profile based on the ratings given to the movies by each user. In this case, the algorithm consist in the weighted mean of the user's ratings and the TF-IDF vector representations of the respective movies. The resulting weighted mean constitute the user profile and is stored in a multidimensional array. To do that, the title of the movie with the rating given by the user were stored in a dataframe. After that the ratings given are standardized by dividing them by 5 that is the maximum rating. Once those preprocessing is completed, with a np.dot() function, the users weighted rating and the TF-IDF vector representations of the respective movies are multiplied giving as a result the array of the user profile. This means that the profile will have high scores for tags belonging to high rated movies and inverse.

The final task consist in taking the cosine similarity between the user profile vectors and content vectors to find their similarity excluding the movies the user has already seen and show the top 10 scores to the user.

5 Results

The results of the project is mainly the relation between the ratings given by the user to the films seen and the list of movies that the system will recommend. This can be done by taking he cosine similarity between the user profile vectors and content vectors to find their similarity excluding the movies the user has already seen.

The results are overall pretty satisfactory. Here I report the ratings given by *"user 50"* filtering only the movies rated more or equal than 4 by the user:

```
df_user_data_with_tags[df_user_data_with_tags.rating>=4]
```

	index	movieId	title	genres	rating_mean	rating_median	num_ratings	stemmed	movie_tags	userId	rating	timestamp	weight
0	106	110	Braveheart (1995)	Action|Drama|War	4.008481	4.0	68803.0	good versus evil best war film great 18th cent...	179 521 1007 258 340 20 285 814 887 186 877 16...	50	5.0	1005875408	1.0
1	248	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Sci-Fi	4.120455	4.0	81815.0	good versus evil scifi cult great clone storyt...	179 1007 796 258 340 864 909 20 285 814 887 18...	50	5.0	1005875323	1.0
2	823	912	Casablanca (1942)	Drama|Romance	4.210098	4.5	31095.0	spi good versus evil best war film dialogue dr...	179 388 1007 258 340 864 814 285 28 186 445 73...	50	4.0	1005875337	0.8
3	1249	1380	Grease (1978)	Comedy|Musical|Romance	3.307229	3.5	19077.0	great storytel teenag honest based on a play b...	179 521 779 192 285 186 445 833 223 449 853 89...	50	4.0	1005875421	0.8
4	1532	1721	Titanic (1997)	Drama|Romance	3.405709	3.5	44787.0	great storytel entirely dialogu honest scifi g...	179 521 779 340 192 20 285 814 887 186 445 732...	50	5.0	1005875430	1.0
5	1733	1968	Breakfast Club, The (1985)	Comedy|Drama	3.827884	4.0	27894.0	dialogue driven great storytel teenag entirely...	179 1037 1007 258 340 864 814 285 887 186 160 ...	50	4.0	1005875496	0.8
6	2495	2779	Heaven Can Wait (1978)	Comedy	3.536849	3.5	2456.0	memori great storytel based on a play life phi...	179 1007 864 814 285 28 186 445 732 833 449 84...	50	5.0	1005883827	1.0
8	2932	3261	Singles (1992)	Comedy|Drama|Romance	3.433502	3.5	2970.0	great storytel entirely dialogu honest great d...	179 661 1007 258 340 779 192 285 186 160 445 8...	50	4.0	1005875445	0.8

Figure 6: Movies rated by the user

And now, after performing the recommendation algorithm described before, and taking the similarities, the recommender suggests the following movies:

	title	movieId	genres	rating_mean	rating_median	num_ratings	stemmed	movie_tags
0	Jerry Maguire (1996)	1393	Drama|Romance	3.603809	4.0	25884.0	great storytel entirely dialogu honest good st...	179 1007 779 340 192 814 285 887 266 186 445 7...
1	Pride of the Yankees, The (1942)	5644	Drama	3.817822	4.0	505.0	good versus evil great storytel movielens top ...	179 258 864 20 285 814 149 186 160 445 732 833...
2	Forrest Gump (1994)	356	Comedy|Drama|Romance|War	4.056585	4.0	97040.0	great storytel oscar (best music - original sc...	179 627 1007 192 20 285 814 887 186 160 445 73...
3	Stand by Me (1986)	1259	Adventure|Drama	3.986363	4.0	25115.0	memori great storytel teenag movielens top pic...	179 1007 258 864 814 285 887 186 160 445 732 4...
4	Gladiator (2000)	3578	Action|Adventure|Drama	3.956335	4.0	48666.0	good versus evil best war film great storytel ...	179 1007 779 340 864 20 285 814 887 186 877 16...
5	Rocky (1976)	1954	Drama	3.726224	4.0	18464.0	great storytel weapon honest gritti superhero ...	179 367 258 779 20 285 814 887 186 160 445 732...
6	Natural, The (1984)	3098	Drama	3.789702	4.0	5535.0	good versus evil great storytel honest great d...	179 1007 258 779 864 814 285 887 186 160 445 7...
7	Glory (1989)	1242	Drama|War	3.933675	4.0	13871.0	best war film great storytel weapon entirely d...	179 627 1007 805 258 340 20 285 814 149 186 16...
8	Rush (2013)	104913	Action|Drama	3.869772	4.0	3467.0	great storytel weapon entirely dialogu honest ...	179 1007 258 340 20 285 814 149 186 508 160 44...
9	Voices from the List (2004)	86504	Documentary	4.124444	4.5	1800.0	best war film great storytel us histori entire...	179 1007 258 340 864 1005 814 285 887 28 64 18...

Figure 7: Movies recommended to the user

As can be seen comparing the two tables, we got pretty good results. In fact there are some highly rated films as *Braveheart (1995)*, *Star Wars: Episode IV - A New Hope (1977)*, and *Casablanca (1942)* that show the tag: *"good versus evil"*. This tag is present also in some recommendation provided by the system such as: *Pride of the Yankees (1942)* ranked 2^{nd}, *Gladiator (2000)* ranked 5^{th}, and *The Natural (1984)* ranked 7^{th}.

Another interesting fact is that without taking into account the genres of the films in the recommendation system, it recognized the attitude of the user in liking drama and romance films and recommended almost all the films of that genre.

6 Conclusions

Content-based filtering is a great approach for a recommendation system but it has also its drawbacks. On one hand *"content representations"* are varied and they expand the options to use different approaches like: text processing techniques, the use of semantic information, inferences and many other, therefore it is easier to make more transparent systems since the same content is used to explain the recommendation. On the other end, content-based systems tend to recommend items similar to those already rated by the user with a tendency of over-specialization. This is a problem when the user likes two things that are very different among each other. In this case, more sophisticated machine learning techniques would address the issue.

In future works could be interesting to constitute the vector representation of the movies also taking into consideration the year of production and see if this could optimize the efficacy of the recommender. Interesting would be also to deal with a dataset that provide entire description/feedback for each movie and therefore, apply more sophisticated NLP techniques.