Movie Recommendation System Using Natural Language Processing

Jai Soni
Northeastern University
soni.j@husky.neu.edu

ABSTRACT

In the world of customization, and hundreds of products available as substitutes, recommendation systems are playing central role in making higher profits. Building a recommender that provides personalized results on user driven inputs and no prior user-data is still expanding in research. We propose a method to use latent semantic similarity and cosine similarity on movie synopsis and movies metadata to get similar content and integrate this technique of finding similarity with other content based filtering, collaborative filtering and hybrid recommendation systems to get better personalized results.

1. INTRODUCTION

A recommender system is a technology that is deployed in the environment where **items** (products, movies, events, articles) are to be recommended to **users** (customers, visitors, app users, readers) or the opposite. Typically, there are many items and many users present in the environment making the problem hard and expensive to solve. Imagine a shop. Good merchant knows personal preferences of customers. Her/his high-quality recommendations make customers satisfied and increase profits. In case of online marketing and shopping, personal recommendations can be generated by an artificial merchant: the recommender system. [1]

The research around recommendation systems has been going on for several decades now, but the interest remains high because of the abundance of practical applications and the problem rich domain. Many such online recommendation systems implemented and used are the recommendation system for books at Amazon.com, for movies at MovieLens.org, CDs at CDNow.com etc. Recommender Systems have added to the economy of the some of the e-commerce websites (like Amazon.com) and Netflix which have made these systems a salient part of their websites.

A glimpse of the profit of some websites is shown in table below:

Netflix	2/3rd of the movies watched are recommended
Google News	recommendations generate 38% more click-troughs
Amazon	35% sales from recommendations
Choice stream	28% of the people would buy more music if they found what they liked

Table 1. Companies benefit through recommendation system

Recommender Systems generate recommendations; the user may accept them according to their choice and may also provide, immediately or at a next stage, an implicit or explicit feedback. The actions of the users and their feedbacks can be stored in the recommender database and may be used for generating new recommendations in the next user-system interactions. The economic potential of theses recommender systems has led some of the biggest e-commerce websites (like Amazon.com, snapdeal.com) and the online movie rental company Netflix to make these systems a salient part of their websites. High quality personalized recommendations add another dimension to user experience. The web personalized recommendation systems are recently applied to provide different types of customized information to their respective users. These systems can be applied in various types of applications and are very common now a day. [2]

We can classify the recommender systems in two broad categories:

1. Content-based filtering approach
2. Collaborative filtering approach

Content-based Filtering

Content-based filtering methods are based on a description of the item and a profile of the user's preferences. In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes. In other words, these algorithms try to recommend items that are like those that a user liked in the past (or is examining in the present). Various candidate items are compared with items previously rated by the user and the best-matching items are recommended. This approach has its roots in information retrieval and information filtering research. [3]

Collaborative Filtering

Collaborative filtering system recommends items based on similarity measures between users and/or items. The system recommends those items that are preferred by similar kind of users. Advantages of collaborative filtering are:

1. It is content-independent i.e. it relies on connections only
2. Since in CF people makes explicit ratings so real quality assessment of items are done.
3. It provides serendipitous recommendations because recommendations are based on user's similarity rather than item's similarity [1]

Figure 1 Clearly explains the differences in these two methods.

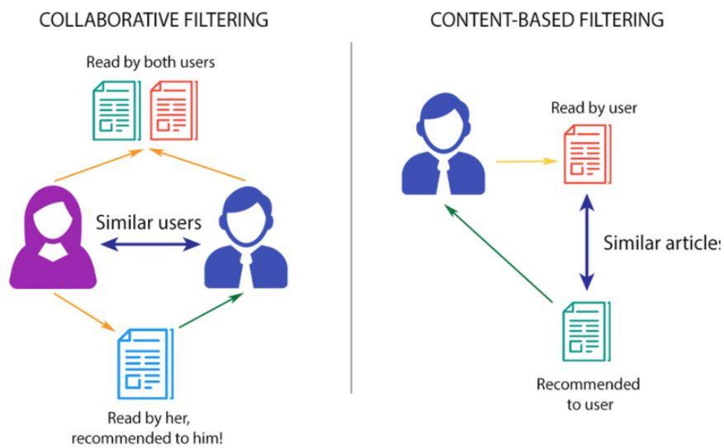

Figure 1. Collaborative Filtering Vs. Content Based Filtering. [4]

2. BACKGROUND

Recommendation systems are commonly seen in places where a user shops for a product or browses around looking for something like on YouTube. These recommendations systems use hybrid systems that involve both content based and collaborative filtering approaches. Since building a Movie recommendation system is one of the first steps in getting started with understanding of Recommendation systems, many people have contributed on building and improving recommendation systems for Movies. A good guide to start building your own Movies recommendation system is mentioned by Author Rounak Banik [5]. On our project, the focus lies on improving a content based recommendation system using Natural language processing techniques such as Tokenization, Lemmatization, Stemming etc.

3. DATA SOURCES

The dataset is taken from Group Lens, which has collected Movie Lens's information for movies, ratings and user's dataset sharing platform. Since the movie lens's large dataset contains additional data for metadata, credits and keywords, I have used Large dataset containing more than 45,000 movies as well as movie lens's Small dataset for movies containing more than 9000 movies [6] over which we have built our recommendation systems due to limited computation power.

4. METHODS

4.1 Simple Genre Based Recommender

Based on IMDB's weighted rating formula, the weighted rating is calculated for each movie based on Vote average (Average Rating) and Vote Counts (Number of Ratings) movies. The formula is described as follows:

Weighted Rating (WR) = (v/(v+m)) R+(m/(v+m)) C

Where,
R = average for the movie (mean) = (Rating)
v = number of votes for the movie = (votes)
m = minimum votes required to be listed in the
C = the mean vote across the whole report [7]

After calculating the weighted rating, we take a required genre and get the top movies with vote count over 85 percentiles. After getting these top movies for the genre we provide the first 100 results sorted by weighted rating which we calculated earlier.

Figure 2. below shows the result for top animation movies using this recommender.

title	year	vote_count	vote_average	popularity	wr
The Lion King	1994	5520	8	21.6058	7.592229
Spirited Away	2001	3968	8	41.0489	7.481030
Howl's Moving Castle	2004	2049	8	16.136	7.217022
Princess Mononoke	1997	2041	8	17.1667	7.215358
My Neighbor Totoro	1988	1730	8	13.5073	7.144693
Up	2009	7048	7	19.3309	6.859731
Inside Out	2015	6737	7	23.9856	6.854574
Despicable Me	2010	6595	7	22.2745	6.852092
WALL·E	2008	6439	7	16.0884	6.849265
Finding Nemo	2003	6292	7	25.4978	6.846500

Figure 2.Result for top Animation movies

Although the results are interesting, the major drawback of using this system is that it provides the same recommendation for everyone, as it just depends on weighted ratings.

4.2 Content Based Recommender System

We have built four content based recommendation systems using the movie's content. For cleaning and filtering these contents we have used the following Natural Language Processing techniques:

Removing Stopwords and Punctuations:
We have used pythons NLTK library to remove words that don't provide meaning to a sentence for computer. Words like 'a, but, how, the, or, what' etc. are referred to as stopwords.
We also removed all the punctuations occurring in the sentence. Removing stopwords and punctuations help in reducing no. of unwanted features thus reducing complexity of the machine learning model.

Tokenization
Tokenization is the process of breaking up the given text into units called tokens. The tokens may be words or number or punctuation mark. Tokenization does this task by locating word boundaries. Ending point of a word and beginning of the next word is called word boundaries. Tokenization is also known as word segmentation. [8]

Stemming
The idea of stemming is a sort of normalizing method. Many variations of words carry the same meaning, other than when tense is involved. The reason why we stem is to shorten the lookup, and normalize sentences. 'I was taking a ride in the car' and 'I was riding in the car' mean the same when tense is not taken into the picture. [9] So we remove the tenses of the word.

We are using nltk's snowball stemmer for this purpose.

Lemmatization

Like Stemming, in computational linguistics, lemmatisation is the algorithmic process of determining the lemma for a given word. Since the process may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence (requiring, for example, knowledge of the grammar of a language) it can be a hard task to implement a lemmatiser for a new language. [10] We are using nltk's wordnet lemmatizer to do this job for us.

Count Vectorization

It is a technique to Convert a collection of text documents to a matrix of token counts. We are using sklearn's Count Vectorizer for creating a count against vocabulary features from our text data.

TF-IDF

In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document and is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. [11] We are using sklearn's TFIDFVectorizer for getting the most important word.

Latent Semantic Analysis

Latent semantic analysis (**LSA**) is a technique in natural language processing, distributional semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text (the distributional hypothesis). A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of rows while preserving the similarity structure among columns. Words are then compared by taking the cosine of the angle between the two vectors (or the dot product between the normalizations of the two vectors) formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words. [12] We used UMBC's api service to get the sentence to sentence similarity score. [13]

4.2.1 Movie Description based Recommender-1

To start with our journey of providing personalized recommendation, we first gathered the list of movies that the user has already watched. For these movies we took the movie's overview from the dataset and passed it through the text processing function that cleaned the sentences following the text cleaning techniques mentioned above. Then we combined the overview for all the movies to make a single sentence. We then used UMBC's api to get the LSA similarity score between the sentence we generated and the text processed overview of the movies that the user has not watched. After getting the similarity score, we sort the movies in the descending order and provide it to the user.

Figure 3 and 4 shows the results for movies recommended using this technique for user id 1 and user id 100 respectively.

title	similarity	vote_count	vote_average
Dumbo	0.587255	1206	6
Epic	0.365386	1143	6
The Wizard of Oz	0.362549	1689	7
Snow White and the Huntsman	0.361434	3183	5
Dumb and Dumber To	0.361130	1140	5
TRON: Legacy	0.356697	2895	6
Oldboy	0.340289	2000	8
Chronicle	0.340005	1965	6
Rise of the Guardians	0.337160	1981	7
Dracula Untold	0.333092	2439	6

Figure 3. Recommendations for user id 1

title	similarity	vote_count	vote_average
Fargo	0.622953	2080	7
Twelve Monkeys	0.619132	2470	7
Independence Day	0.589538	3334	6
Toy Story	0.589521	5415	7
The Rock	0.585872	1474	6
Mission: Impossible	0.572218	2677	6
Heat	0.572186	1886	7
True Lies	0.478509	1138	6
Man of Steel	0.446365	6462	6
The Wizard of Oz	0.445946	1689	7

Figure 4. Recommendations for user 100

Since the recommendations are provided different for different user's, this is a good way of recommending the movies for a user. But this technique has a limitation. Calculating the latent similarity takes longer time when we have users who have watched many movies. As we have longer sentences and more text features to compare, getting similarity scores from the UMBC's simservice api took long time.

4.2.2 Movie Description based Recommender-2

Since one kind of recommendation system is never enough to provide a variety of better recommendations. We explored ways to find recommendations for similar movies when a movie is provided as an input by the user.

We also looked for an alternative way to get the similarity measure and got **Cosine similarity** technique as the substitute. The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them. This metric is a measurement of orientation and not magnitude, it be a comparison between documents on a normalized space because we're not taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. What we must do to build the cosine similarity equation is to solve the equation of the dot product for the cosθ:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

And that is it, this is the cosine similarity formula. Cosine Similarity will generate a metric that says how related are two documents by looking at the angle instead of magnitude. [14]

To provide more weight to the movies with similar taglines, we added the tagline feature from the movies dataset and added it to a placeholder column called description. We then added passed it to our recommendation model that calculates the cosine similarity between the description and provides a similarity score. Again, we provide the top similar movies for a movie based on highest similarity scores. Figure 5 shows the recommendation for a movie called 'Star Wars'.

```
desc_based_recommendation('Star Wars').head(10)
```

```
949          The Empire Strikes Back
962               Return of the Jedi
8755    Star Wars: The Force Awakens
6690                 Shrek the Third
7539              Shrek Forever After
4815               Where Eagles Dare
2890                  Shanghai Noon
5383                         Shrek 2
5088              The Thief of Bagdad
279                 The Swan Princess
Name: title, dtype: object
```

Figure 5. Recommendation for movies like 'Star Wars'

We can get similar movies to Star Wars like the Empire strikes back, Return of the Jedi and Star Wars the force awakens, when we have not provided even the title or genre. But we also get movie like Shrek Forever after and The Swan Princess which are of completely different Genre than Star Wars.

4.2.3 Metadata Based Recommender

To get even better results in our recommendation engine, we added features like Keywords(tags), Cast and Crew of the movie. We added the top 3 movie actors from the movie, and added Directors name 3 times to increase the weightage when getting similarity, to our bowl. We also added genre of the movie to get movies with similar genres. We used the same recommendations engine to get the predictions. Figure 6. Shows the result for 'Star Wars'.

```
desc_based_recommendation('Star Wars').head(10)
```

```
2120        Star Wars: Episode I - The Phantom Menace
4137     Star Wars: Episode II - Attack of the Clones
6199     Star Wars: Episode III - Revenge of the Sith
970                           The Empire Strikes Back
4789                                         THX 1138
983                                Return of the Jedi
2705                                American Graffiti
7088                           Star Wars: The Clone Wars
8865                        Star Wars: The Force Awakens
8079                  Journey 2: The Mysterious Island
Name: title, dtype: object
```

Figure 6. Recommendations for movies like 'Star Wars'

We felt that the predictions are much better now, but we also got the movie 'THX1138' in our list which is not amongst the most popular movies that are listed for 'Star Wars'.

Figure 6. shows that the adding of weights to the director works as we get most of the Christopher Nolan movies when we search for the movie 'Inception'

```
desc_based_recommendation('Inception').head(10)
```

```
6623                              The Prestige
3381                                   Memento
4145                                  Insomnia
2085                                 Following
8031                     The Dark Knight Rises
8613                              Interstellar
6981                           The Dark Knight
6218                             Batman Begins
5638     Sky Captain and the World of Tomorrow
8500                                   Don Jon
Name: title, dtype: object
```

Figure 7. Recommendation for movies like 'Inception'

4.2.4 Popularity Based Recommender

Since our current recommender doesn't take popularity and ratings into account, it shows movies like 'THX1138' over many other popular movies when we searched for 'Star Wars'. We improved our recommendation system by returning only popular movies with more number of ratings.

Figure 8. Shows the improved recommendation for 'Star Wars'

title	vote_count	vote_average	year	wr
The Empire Strikes Back	5998	8	1980	7.671623
Star Wars: The Force Awakens	7993	7	2015	6.866624
Return of the Jedi	4763	7	1983	6.793425
Star Wars: Episode III - Revenge of the Sith	4200	7	2005	6.771573
Iron Man 2	6969	6	2010	5.988459
Divergent	4784	6	2014	5.984190
Star Wars: Episode I - The Phantom Menace	4526	6	1999	5.983468
X-Men	4172	6	2000	5.982363

Figure 8. Recommendations for movies like 'Star Wars'

4.3 Collaborative Filtering

The Results from our popularity based recommender are impressive, we get most of the similar movies when querying for a movie. While content based are good when we have good amount of content for the movie like the name of actors, movie synopsis, director's information etc. we always don't have all the information required for making relevant recommendations. Also, while we tried to derive user's taste by using movies overview and taglines as input to our model, the recommendations provided by a collaborative filtering model are way better than a content based model. Another advantage of using a collaborative filtering model over Content based model is that it doesn't require any data related to movies content. We have built a CF model using Scikit learn's Surprise library which provides a simple data ingestion for making recommendations through CF. It also provides powerful algorithms like Singular Value Decomposition(SVD) to minimize RMSE and provide great recommendations.

```
#Provide userId, movieId and True Rating
svd.predict(1, 302, 3)
```

Figure 9. Function to get predicted rating for user id-1, movie id-302, true rating-3

Prediction(uid=1, iid=302, r_ui=3, est=2.7450032112214804, details={'was_impossible': False})*For movie with ID 302, we get an estimated prediction of 2.745. One startling feature of this recommender system is that it doesn't care what the movie is (or what it contains). It works purely based on an assigned movie ID and tries to predict ratings based on how the other users have predicted the movie. [15]*

4.4 Hybrid System

Hybrid Recommender leverages the best of both Content based and collaborative filtering techniques. Using ideas from Content based engine and Collaborative filtering based engine, we created a Hybrid recommender system which provided more personalized recommendations for users.

```
hybrid(1, 'Avatar')
```

	title	vote_count	year	id	est rating
1011	The Terminator	4208.0	1984	218	3.113663
522	Terminator 2: Judgment Day	4274.0	1991	280	3.099023
974	Aliens	3282.0	1986	679	2.979387
8658	X-Men: Days of Future Past	6155.0	2014	127585	2.894395
8401	Star Trek Into Darkness	4479.0	2013	54138	2.874042
2014	Fantastic Planet	140.0	1973	16306	2.820920
344	True Lies	1138.0	1994	36955	2.779683
1668	Return from Witch Mountain	38.0	1978	14822	2.747166
1621	Darby O'Gill and the Little People	35.0	1959	18887	2.741992
4017	Hawk the Slayer	13.0	1980	25628	2.699138

Figure 10. Recommended movies for user 1, like movie 'Avatar'

```
hybrid(500, 'Avatar')
```

	title	vote_count	year	id	est rating
2014	Fantastic Planet	140.0	1973	16306	3.245734
4966	Hercules in New York	63.0	1969	5227	3.239042
8658	X-Men: Days of Future Past	6155.0	2014	127585	3.180482
8401	Star Trek Into Darkness	4479.0	2013	54138	3.174624
8419	Man of Steel	6462.0	2013	49521	3.148933
1011	The Terminator	4208.0	1984	218	3.141192
1621	Darby O'Gill and the Little People	35.0	1959	18887	3.102287
1376	Titanic	7770.0	1997	597	3.077261
2132	Superman II	642.0	1980	8536	2.981782
1668	Return from Witch Mountain	38.0	1978	14822	2.975746

Figure 11. Recommended movies for user 500, like movie 'Avatar'

Figure 10. and Figure 11. Shows how we achieved personalized rating similar to 'Avatar' for user Id 1 and user Id 500.

5. RESULTS

As we built different recommendation systems, we improved our recommendation on each recommendation engine we have built using various techniques. Our major focus included exploring Natural Language processing techniques to provide cleaner text as input features for content based recommendation models. We also explored Scikit learns surprise library to build a CF based recommender and Hybrid Recommender systems. These models are of course not a match for industry level recommendation systems, but they are a good way for starting with them.

.

6. CONCLUSION

In this project, I have created 6 types of movie recommender systems:

Simple Genre Based Recommendation System:

We created Top Movies Charts based on Genre and utilized IMDB's Weighted Rating System to calculate ratings which was used to then sort and return top movies.

Content Based Recommendation System: We built four content based recommendation engines

First, we gathered movie's overviews which a user has already seen and rated above average, then we used latent semantic similarity to get the similarity score and created a recommender that provides most similar story to user's liking.

On our second approach on creating taste based recommendation by using NLP techniques used for above, and added tagline to the description as an input

Next, we considered metadata such as cast, crew, genre and keywords as input features to our Recommendation Engine, we also added weights features like director to get more similar results

We then improved our prediction by adding a popularity and ratings filter so that recommendations are given on popular movies

Collaborative Filtering Recommendation System:

We used the powerful Surprise Library to build a collaborative filter based on single value decomposition(SVD). The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie.

Hybrid Recommendation System:

Hybrid Recommender leverages the best of both Content based and collaborative filtering techniques. Using ideas from Content based engine and Collaborative filtering based engine, we created a Hybrid recommender system which provided more personalized recommendations for users.

7. CODE AND DOCUMENTATION

The code and the dataset is available on GitHub along with steps on how to run the code, the link is mentioned below:

https://github.com/jaisoni17/Movie-Recommendation-System

8. REFERENCES

[1] Kordik, Pavel,https://medium.com/recombee-blog/recommender-systems-explained-d98e8221f468

[2] Manoj Kumar, D.K. Yadav, Ankur Singh, Vijay Kr. Gupta. International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3, August 2015

[3] https://en.wikipedia.org/wiki/Recommender_system

[4] Recommender System for News Articles using Supervised Learning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/Collaborative-Vs-Content-Based-Filtering_fig3_318129942 [accessed 21 Apr, 2018]

[5] Banik, Rounak, https://www.datacamp.com/community/tutorials/recommender-systems-python

[6] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872 , Link for dataset https://grouplens.org/datasets/movielens/

[7] https://math.stackexchange.com/questions/169032/understanding-the-imdb-weighted-rating-function-for-usage-on-my-own-website

[8] http://language.worldofcomputing.net/category/tokenization

[9] https://pythonprogramming.net/stemming-nltk-tutorial/

[10] http://textminingonline.com/dive-into-nltk-part-iv-stemming-and-lemmatization

[11] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[12] Susan T. Dumais (2005). "Latent Semantic Analysis". *Annual Review of Information Science and Technology*. 38: 188–230. doi:10.1002/aris.1440380105.

[13] Lushan Han, Abhay L. Kashyap, Tim Finin, James Mayfield and Johnathan Weese, UMBC_EBIQUITY-CORE: Semantic Textual Similarity Systems, Proc. 2nd Joint Conf. on Lexical and Computational Semantics, Association for Computational Linguistics, June 2013.

[14]Christian S. Perone http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[15] Banik, Rounak https://github.com/rounakbanik/movies/blob/master/movies_recommender.ipynb