Robert Trout
05/12/2018
Ling 165
Final Project

Using TF-IDF and SVD for Movie Recommendation

I. Overview

Movie recommendations are an important part of the modern movie watching experience. With the sheer glut of movies to watch, there must be some sort of mechanism to filter the pool of movies that are available to a smaller batch of those that one would enjoy watching. One can lament the fact that blockbuster is out of business but the fact remains, those days are not returning. With the lack of human-to-human contact, automated systems need to be developed to offer those recommendations to the customer.

There are many methods to approach this problem and each comes with its own host of benefits and drawbacks. One could use a recommendation system built on user ratings, which pushes higher rated movies to the top, which will promote well-made movies, but it will lack any fine-tuning. Selections will have to be segregated to genre leaders, which means that the end user must know what type of movie they want to watch. These will also fail when an individual does not realize there is a cross-genre movie that could appeal to their tastes.

One could implement a neural network based on the personal choices of what the individual user enjoys. This would solve some of the issues that plague the user-review method above: recommendations would be personalized and some surprising recommendations could be served to the user that they would never have thought of themselves. One large drawback to this method is that training time in extensive and a history of movie watching patterns is needed before the system can “fine-tine” the recommendations to match what the user would think suits their style.

The TF-IDF and SVD method that is explored in this project answers some of the issues addressed by these other methods and offers a low-cost method for a service that does not have a lot of data on users. This system creates a list of recommendations solely from plot details from the current movie being watched and therefore first-time users or users who do not want to create a profile can be offered other products. It can also reach across genres, as every movie in the database is compared to make a suggestion, and if another movie from a different genre fits, it can easily be recommended.

There are shortcomings, however. All movies must be annotated with plot summaries and the focus can be narrow if only plots are considered as input for the recommendation system. These both will be addressed in the “Further Work” section, but for this current implementation, only plots will be considered for input.

II. Other work

This project relies heavily on one paper for inspiration and guidance with the classification system. This is titled “Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems” by Sonia Bergamaschi and Laura Po at University of Modena and Reggio Emilia. This paper compares two systems of classification, LDA (Latent Dirichlet allocation) and LSA (Latent Semantic Analysis), with the IMDB (Internet Movie DataBase) recommender engine. They find that LSA is a strong contender for a recommender system while LDA does not perform well. Many parameters from this study are used in this project, while sources of summaries are different.

This project is also indebted to two web resources for help with understanding web scraping and working with the sklearn modules. Respectively they are the Wikipedia scraper by Alexander Holt on his github gist: https://gist.github.com/alexanderholt, and a tutorial regarding document clustering by Brandon Rose: http://brandonrose.org/clustering.

III. Outline of Work

There are three main parts of this process and they can be categorized as follows: Data collection, Matrix creation, and Cosine Similarity.

A) Data Collection

Data collection was by far the most time-intensive part of this project as the scrapper had to traverse many pages and collect a whole corpus single-handedly. While *Comparing LDA and LSA* used a database from IMDB, the author of this project decided to collect information from Wikipedia. Both corpora must deal with unreliable user-created data, but Wikipedia is a more heavily trafficked site and has the possibility of a greater amount of detail being generated for each movie title through a “wisdom of the masses” approach to data creation.

American movies were chosen as the focus, for two reasons. One, the shared “Hollywood” culture between those movies will offer a better comparison of plots and two, the recognizable quality of the movies will lead to a recommendation list that will have an easier quality judgement. The author will be able to discern quickly whether good recommendations are being offered and any larger-scale assessment will have a greater likelihood of being able to be judged.

At the onset of this project, more data than just plot was collected, namely genre and year of debut. However, these were not brought into the creation of the matrix and will be addressed in the “Further Work” section later.

B) Matrix creation

After a folder was populated with files containing the plot of each film and title, these where then processed into a tf-idf matrix using the sklearn TfidfVectorizer method. For this project, English stop words were used from the NLTK stopwords list. Punctuation was filtered from the plots and all words stemmed. The NLTK English Snowball stemmer was used to create the stems. Lemmatization was not used as both did not work together, however, that option is considered in "Further Work."

After tokenizing and stemming all words, the tf-idf made a matrix of about 18,585 by 81,176. That corresponds to 18k movies and a vocabulary of 81k words.

Once this matrix was made, this was also applied to the test plots which were to be compared.

The tf-idf matrix was the transformed into a truncated SVD and LDA matrix, also using sklearn's TruncatedSVD and LatentDirichletAllocation method under the decomposition module.

These respectively created matrices of 18,585 by 1000 and 18,585 by 300. These were both values gained from experimentations and recommendation from *Comparing LDA and LSA* (which featured 500 for LSA and 50 for LDA).

Like tf-idf, these transformations are applied to test plots to transform them into a comparable vector.

C) Cosine Similarity

Sklearn also has a built-in cosine similarity function, which the author used for the plain tf-idf comparison, but a custom-built cosine similarity function was used for the Truncated-SVD and LDA comparison.

Once these distances are computed, the closest five movies are then printed and used for recommendations purposes.

IV. Results

In Table 1 and 2 below, the reader will find the results from two test movies, one that is part of a larger "cinematic universe" and another with a stand-alone plot. Any plot can be used, and Table 3 features a movie recommendations based a plot of book. One item to note is that in both tf-idf and Truncated SVD, the film under consideration is the first recommended, which is to be expected.

The fact that LDA only recommends the film under consideration later shows that it has oversimplified the matrix and cannot be considered a strong contender for a recommendation system.

The top ten movies of all time, as per Rotten Tomatoes, were chosen to review in a survey, in order to ensure a strong likelihood of familiarity of survey-takers. The movie recommendation (result 1-5) was randomized. Survey-takers were asked to rate the recommendations as either "good," "bad," or "I don't know." All three methods, vanilla tf-idf, Truncated SVD, and LDA were compared. A baseline of IMDB recommendations were also compiled. The movie under consideration was removed as a potential recommendation, to prevent confusion among survey-takers. Figure 1 shows that breakdown. Due to lack of funds, only a small set of responses were able to be generated, and the answers cannot be considered representative of a completed picture. One takeaway from the survey results is that the survey takers (from Amazon's Mechanical Turk) considered most recommendations good, no matter the system, but IMDB did the best. According to their answers, LDA preformed just as well as LSA, if not better, but as the reader can clearly see, more survey results would be needed before that conclusion could actually be reached. The author of this project will wait until more funds materialize to move forward with more survey data.

Vanilla TF-IDF	Truncated SVD	LDA
Movie 0: The Avengers (2012 film).	Movie 0: The Avengers (2012 film).	Movie 0: Rings (2016 film).
Movie 1: Thor (film).	Movie 1: Avengers: Age of Ultron.	Movie 1: Rings (2017 film).
Movie 2: Son of the Mask.	Movie 2: Thor: Ragnarok.	Movie 2: Golden Rule Kate.
Movie 3: Thor: Ragnarok.	Movie 3: Thor (film).	Movie 3: Comes a Horseman.
Movie 4: Prisoners (2013 film).	Movie 4: Iron Man 2.	Movie 4: The Avengers (2012 film).
Movie 5: Avengers: Age of Ultron.	Movie 5: Iron Man (2008 film).	Movie 5: Tumbleweeds (1999 film).

Table 1 - Avengers (2012 film)

Vanilla TF-IDF	Truncated SVD	LDA
Movie 0: Citizen Kane.	Movie 0: Citizen Kane.	Movie 0: The Cool and the Crazy.
Movie 1: High Noon.	Movie 1: The Ninth Configuration.	Movie 1: Soup to Nuts.
Movie 2: Saboteur (film).	Movie 2: Saboteur (film).	Movie 2: Pacific Rim (film).
Movie 3: The Ninth Configuration.	Movie 3: High Noon.	Movie 3: One Body Too Many.
Movie 4: Young Billy Young.	Movie 4: Armed and Dangerous (1986 film).	Movie 4: Along Came Polly.
Movie 5: An Eye for an Eye (1981 film).	Movie 5: An Eye for an Eye (1981 film).	Movie 5: All American (film).

Table 2- Citizen Kane

Vanilla TF-IDF	Truncated SVD	LDA
Movie 0: A Tale of Two Cities (1935 film).	Movie 0: A Tale of Two Cities (1935 film).	Movie 0: Sword in the Desert.
Movie 1: Stealing Beauty.	Movie 1: Hello Again (1987 film).	Movie 1: Coming to America.
Movie 2: 50 First Dates.	Movie 2: Stealing Beauty.	Movie 2: Mary Reilly (film).
Movie 3: Hello Again (1987 film).	Movie 3: Is Paris Burning? (film).	Movie 3: Jeepers Creepers 3.
Movie 4: The Balcony (film).	Movie 4: Broken Blossoms.	Movie 4: Next Time We Love.
Movie 5: Is Paris Burning? (film).	Movie 5: While Paris Sleeps (1932 film).	Movie 5: Minority Report (film).

Table 3- A Tale of Two Cities (novel)

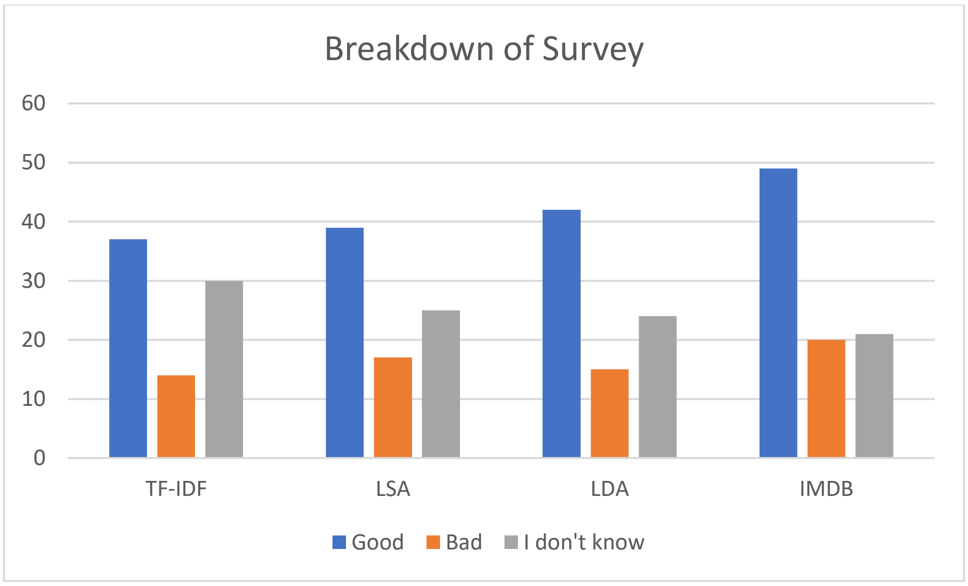

Figure 1

V. Future Work

This system offers functionality that can be upgraded to present a much more versatile system. As mentioned in the introduction, the downside of this system is its myopic focus on plot, but this could be expanded to include more input such as actors, staff, genre, or even year. This could be implemented with a separate matrix to include those details, or include these all in the same matrix. Work would have to be down to see how both types of inputs could be interfaced and weighted properly.

More work could also be done on fine-tuning the parameters, such a min-df and max-df for the tf-idf matrix or different values for the number of features in the Truncated SVD and LDA.

Lemmatization could be used instead of stemming or a different stemmer used instead of the Snowball one used in this project.

A wider base of films could used, with international and tv-show plots as possible candidates.

As a related project, various classification methods could be used to create categories for the various movies to allow for wider “genres” of plots to be recognized.

VI. Conclusion

The Truncated SVD method was the strongest of the recommendation methods created in this project, which the author and survey-takers agree, give “good” results.

The author hopes that his work can be used and expanded to make a system that works robustly in situations that require recommendations in a constrained setting, like lack of internet connectivity or user profiles.

Works Cited

Bergamaschi S., Po L. (2015) Comparing LDA and LSA Topic Models for Content-Based Movie Recommendation Systems. In: Monfort V., Krempels KH. (eds) Web Information Systems and Technologies. Lecture Notes in Business Information Processing, vol 226. Springer, Cham

Rose, Brandon. (n.d.). Document Clustering with Python. Retrieved May 12, 2018, from http://brandonrose.org/clustering

Holt, Alexander. (December 17, 2018) Wikipedia Scape List. Retrieved May 12, 2018, from https://gist.github.com/alexanderholt