ANALYSIS OF TOP RATED MOVIES ON IMDB USING K-MEANS AND LATENT DIRICHLET ALLOCATION ALGORITHMS

A PREPRINT

Beiping Qiu
Mechanical Engineering and Materials Science Department
Duke University

Naman Jain
Computer Science Department
Duke University

April 27, 2019

ABSTRACT

Grouping and clustering algorithms are very useful in today's world. They help us to form an association between a lot of seemingly unrelated things. But the question is whether all of them are as effective as the other or some are better. In this paper, we take a dataset of movies which contains plot summaries and associations are formed, i.e., clustering is done on the basis of some specific keywords found in these plot summaries. We use 2 major grouping algorithms, K-means clustering algorithm and Latent Dirichlet Allocation (LDA) algorithm.Then, we compare the results and see whether they have similar results or is there a huge difference.

1 Introduction and Motivation

We use unsupervised learning models to cluster plots of 100 Top Rated Movies of All Time from IMDb into different groups, visualize the results and identify their latent topics/structures. Under predictive analysis of the NLP approaches, LDA topic modeling and obtaining the percentage distribution of the movies in each of the genres produced the best accuracy, the results were valid and were verified with the K-Means clustering results. We wanted to compare LDA and K-Means so that we can see how the results pan out. Also, we wanted to see which types of movies the public likes in general and maybe our research can be used by budding filmmakers to make movies that appeal more to the audience.

2 Dataset

The dataset used in the project has been made from scratch. We found a list of top movies from IMDb and then we got the corresponding summaries from IMDb. However, these summaries did not provide a large enough dataset to apply the algorithms we intended to apply (Especially, for Latent Dirichlet Allocation). So, we crawled through the Wikipedia pages of all these movies and got the corresponding summaries from there too. Then, we combined both of them to give us a dataset to work on.

3 Methodology

We followed a few steps to make our data more relevant for both our algorithms.

3.1 Tokenizing and Stemming

After getting the data in a combined form of summaries from IMDb and Wikipedia, we used the nltk library in Python to remove the stopwords. Stopwords are basically the ones that are used very often in all types of documents and generally provide support to other words and don't have a significant meaning of their own. For example, auxilliary verbs like 'is' and articles like 'the' are considered as stopwords. We also didn't take into account the numeric and punctuation tokens. After removing the stopwords, the next step was stemming using nltk library. Now, stemming is primarily about finding the word stem of the token at hand. Word stem is the most basic form of a word which is then modulated according to the task at hand. For example, teach is a stem for many words such as teaches, teaching, teacher etc. Getting the stems for all the words helps us to group different words into one category. After stemming, we create a mapping from these word stems to the original words.

3.2 Generation of TF-IDF (term frequency–inverse document frequency) matrix from the tokens and word stems

Term frequency refers to the count of a particular word's occurrence in a given document. If a word has high term frequency, it will be considered a more important word for the document. So, we had to remove the stopwords as they had a very high term frequency but gave no unique meaning to the document.

Inverse document frequency is used to tackle the problem faced due to stopwords, i.e., high term frequency but otherwise, low value in providing a distinct view on the document. Even after removing the stopwords, there would be words that occur in all the documents at hand and will not be helpful in identifying them in a unique way. For example, if the documents we have are all related to a study of flowers, all the documents will have the word 'flower', which is not a stopword but will not help us distinguish these documents. Hence, IDF is used to check the importance of a word based on the fact whether it occurs a lot or just sparsely in other documents.

So, TF-IDF is used to signify the importance of a word in a document, which is part of a corpus. It is commonly used in text-based recommender systems and search engines to rank the relevance of a document, given the query generated by the user's search words. The tf–idf value increases with the increasing count of a particular word in a given document and decreases with the increasing count of the number of documents in the corpus that include that word.

3.3 K-Means Clustering

K-means is a clustering algorithm and here, we apply this algorithm (using sklearn library) to group the movies in the given list into an arbitrary number of clusters (This number can be changed as per requirement and can be controlled in the code). As the titles originally had a rank, we find the average rank for each cluster. Next, we find the cluster centers for all the clusters and get the top-k words in each cluster. On the basis of these words, we see how these movies have been grouped into one cluster. For example, a cluster with words like battle, soldiers and war is different from a cluster with family, love and marriage. We used Principal Component Analysis (PCA) to reduce the dimensionality of the TF-IDF matrix (100x538).

3.4 Latent Dirichlet Allocation (LDA) Implementation

As topic structures in a document are hidden, LDA helps us to decide which word in a document comes from which topic and also tells us what this document is about on the basis of all the words and topics. It can also be used to generate documents on a particular topic, given the requirements (proportion of each topic in the required document). We used LDA to cluster the movies on the basis of their summaries, just like we did with K-Means. Again, the number of clusters (or in this case, topics), can be defined as per requirement. We find the top words from each cluster (topic) and then see how these movies have been grouped.

4 Results

We took top 100 movies from IMDb and generated 3 clusters for both K-means and LDA. The table below shows all the clusters with associated words and movie count. Apparently, all the results of both the algorithms are similar, just the cluster numbers are different. For example, Cluster 0 in LDA is similar to Cluster 2 in K-means and evidently they have related words like police, car, killing, asks etc and it will not be wrong to say that the movies in these 2 clusters are related to detective work and murder investigations. Similarly, Cluster 0 in K-Means is similar to Cluster 1 in LDA and they have words related to family and love. Lastly, Cluster 1 in K-means is similar to Cluster 2 in LDA and we see a lot of similar words like army, soldiers, war etc. Both the clusters contain almost the same words. So, we can say that this cluster represents war movies and the top rated on IMDb are of 3 major types: detective investigations, family movies and war movies.

Cluster	K-Means Movie Count	LDA Movie Count	K-Means Top Words List	LDA Top Words List
0	10	48	family, singing, performed, marries, filmed, dance	days, say, night, asks, car, away, killing, goes, police
1	28	36	army, soldiers, killing, war, commanded, orders, men, general, battle, forcing	friend, family, father, home, love, want, new, life, becomes
2	62	16	police, father, car, killing, home, friend, meet, asks	killing, war, orders, men, soldiers, army, forcing, prison, attacked, captain

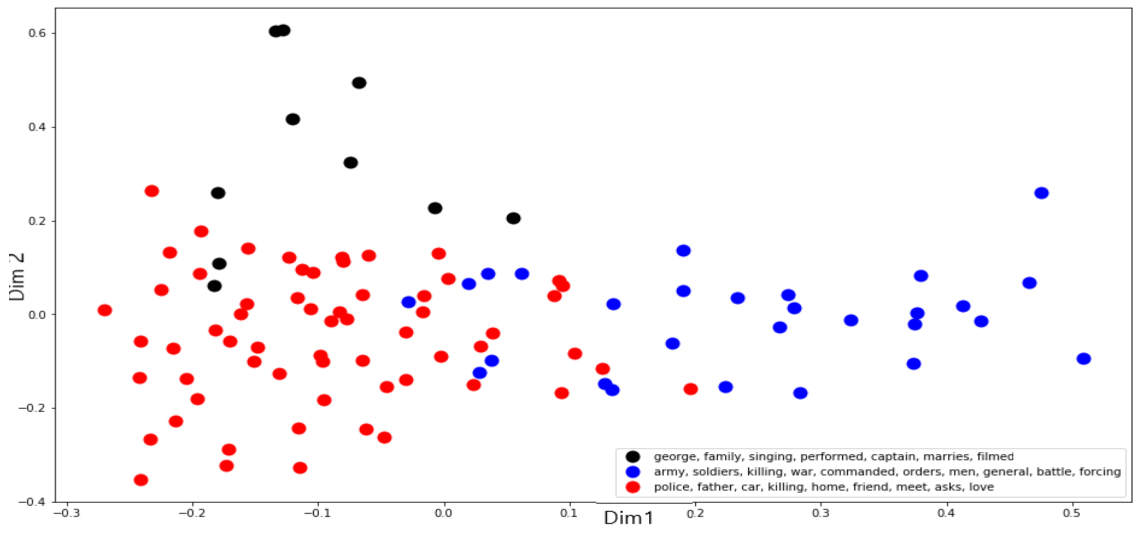

Figure 1: K-means Clustering Results

5 Conclusion

- We clustered unlabeled textual documents into groups and discovered latent semantic structures using Python.
- Preprocessed text by tokenizing, stemming and stopwords removing, and extracted features by term frequency-inverse document frequency (TF-IDF) approach.
- Trained unsupervised learning models of K-Means Clustering and Latent Dirichlet Allocation (LDA). Identified latent topics and keywords of each document for clustering and calculated document similarity.
- We observed that both K-means and LDA give us almost similar cluster, grouping together same types of movies on the basis of occurrence of similar keywords in their plot summaries.
- Visualized model training results by dimensionality reduction using Principal Component Analysis (PCA).
- Only plot summaries from IMDb do not give us enough data to work on.

6 Future Work

- Identifying the emotions portrayed in a scene by carrying out sentiment analysis on dialogues that occur within a certain interval in the movie.
- More movie-related attributes can be added to the dataset and we can see what attribute gives us a better result and which one doesn't.
- We can increase the dataset to a point where we start seeing changes in the results. For example, we can see much better results than the current observation or there could be saturation after a while or maybe it could get worse. By better results, we mean that K-means and LDA give us more similar results.

References

[1] The movie list: `https://www.imdb.com/list/ls055592025/`

[2] `https://medium.com/datadriveninvestor/data-science-analysis-of-movies-released-in-the-cinema-betwee`

[3] `https://medium.com/@ibjects/imdb-dataset-visualization-data-analytics-using-pandas-97b5c6f03c6d`

[4] LDA:-`https://quantdev.ssri.psu.edu/sites/qdev/files/topic_modeling_tutorial-Gutenberg-chapter_as_document_0.html`

[5] `http://www.davidsbatista.net/blog/2017/04/01/document_classification/`

[6] The Little Book of LDA: `https://ldabook.com/`