CS 410 Project Report

Jose Morales (jmoral39@illinois.edu)
Bobby Zahn (lyzahn2@illinois.edu)
Dipali Ranjan (dipalir2@illinois.edu)

Introduction

Our project is intended for users that want to discover new movies they may not have seen yet. More specifically, it will help users find movies that are related to movies they have watched while also maintaining a similar rating. Our project does topic analysis on the top and bottom 250 user rated movies on IMDb (a reliable source to find movie ratings) and generates multiple dynamic graphs. It is useful because it facilitates searching for a new movie by focusing on the visualization of the data, making it easier to analyze. We have published some of these visualizations on a web page for people to explore, which can be seen by opening index.html from the Github repository.

Related Work

While topic modeling has previously been explored for movie recommendations [1], we wanted to expand on this to try to differentiate movies which are well-rated or poorly-rated by users overall. With this in mind, we chose to analyze the top 250 and bottom 250 movies on IMDb based on user ratings, rather than using preexisting movie summary datasets which do not include rating information. In addition, we wanted to add interesting visualizations to help users explore and discover new movies. Many services such as Amazon and Netflix contain movie recommendation systems, but these generally only provide users with lists of recommended movies and do not have any interesting visualizations. Our visualizations were inspired by prior work on visualizing cluster relationships for recommendations [2] and on interactive movie recommendation systems [3].

Implementation/Methodology

Our probabilistic approach employs LDA to model user top and bottom IMDb rated movie summaries. We assume that movie summaries can be modeled as mixtures of topics, and that each topic represents a probability distribution over movies. In this process,we hope to find the similarities in movies which are highly/lowly ranked.

Data Collection:

To collect our data, we scraped the IMDb web sites which contained the top and bottom 250 IMDb rated movie Ids. Next, we wrote a python script that found tags for each of the movies in the lists, allowing us to write the movie IDs in a text file. We then used IMDbPY (Python package) to gather the movie titles, year of release and summaries which were later stored into separate text files.Besides the list of movies collected, we also used the CMU Movie Summary Corpus for topic modeling and visualization. This dataset contains 42,306 movie plot summaries and associated metadata.

Probabilistic Topic Models:

We review topic modeling, focusing on Latent Dirichlet allocation (LDA) (Blei, Ng, and Jordan 2003), which is one of the simplest probabilistic topic models. LDA decomposes a collection of documents into topics—biased probability distributions over terms—and represents each document with a (weighted) subset of the topics. When such a model is fit to a set of documents, the topics represent themes in the corpus.

LDA is a matrix factorization technique. It takes as input a bag of words matrix (i.e., each document represented as a row, with each columns containing the count of words in the corpus). The aim of each algorithm is then to produce 2 smaller matrices; a document to topic matrix and a word to topic matrix that when multiplied together reproduce the bag of words matrix with the lowest error. [4]

1. For K topics, choose each topic distribution β_k (β_k belongs to a multinomial distribution over |V| words in vocabulary is derived from Dirichlet distribution $Dir(\beta_k \eta)$, η is a Dirichlet prior of word distributions over topic.
2. For each document in the corpus:
   (a) Choose a distribution over topics θ_d. (The variable θd is a distribution over K topics is constructed from Dirichlet distribution $Dir(\theta_d|\alpha)$)
   (b) For each word in the document d.
   I. Choose a topic assignment z_{dn} from θ_d. (Each z_n is a number from 1 to K.)
   II. Choose a word w_{dn} from the multinomial topic distribution $\beta_{z_{dn}}$.

Same set of topics is used for every document, but that each document is represented by different proportions θ_d of each topic. [5] LDA doesn't assign any 'hard' topics to a document, for example it models that a corpus with topics "sports", "health" and "business", some articles can be about "sports" and "business", others are about "sports" and "health", and that the topic of "sports" is similar in each.

We use text-processing techniques to enhance the quality of LDA output. We employed the following techniques before using the LDA model.

1. **Stopwords Removal-** are the words we want to filter out before training the model. These are usually high frequency words that aren't giving any additional information to our labeling. In fact, their influence might have a negative effect on the model. In English, they could be 'the', 'is',' which',' and',' on' which may not necessary reflect on the themes of the topic.
2. **Stemming** is the process for reducing inflected words to their word stem,base or root form.For example in our document containing the summary of 'Friday the 13th' , there are featured words 'fearsome' and 'feared' which the LDA model will treat differently with different frequencies. By stemming them, it groups the frequencies of different inflection to just one term — in this case, 'Fear'.
3. **Tokenizing** is the identification of linguistically meaningful units from the surface text. We have used the function to remove white spaces and punctuation marks between the texts and make our summary interpretable to the LDA model by creating a list of words. For example, "unlocks the never-before-seen secret world inside your smartphone" will be transformed to ['unlocks', 'the', 'never', 'before', 'seen', 'secret', 'world', 'inside', 'your', 'smartphone']. We discarded any tokens smaller than size 2.

4. **POS Tags-** The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. We use pre-defined POS tags to remove the proper nouns from the text.
5. **Vectorizer** : It is used to convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts sparse matrix representation. We also use token_pattern which is a string Regular expression denoting what constitutes a "token". In our model it's all alphabets in English language and numeric numbers.
6. **Multidimensionality Scaling (Visualization)** is used to provide a visual representation of similarity between different topics. We have used a 2 dimensional plot to visualize the movies which are similar, based on the distances between the points.
7. **Dimensionality Reduction using t-SNE algorithm (Visualization):** To visualize our highly dimensional data in 3-D space, we use t-Distributed Stochastic Nethe prighbor Embedding (t-SNE). The high dimensional data point will be transformed into different 3-D vectors in different batches, relative to other data points in the batch. [6]

Clustering:
Before clustering, we implemented stopwords removal, stemming, and tokenizing. We then used a vectorizer to convert the data to a tf-idf (term frequency-inverse document frequency) matrix. We defined the distance between two documents as the cosine distance between each

document's vector in the tf-idf matrix, or for two documents u and v.
We used two different clustering methods:

1. **K-means clustering** initializes the centroids of some number of clusters k and assigns each data point to its nearest cluster. Then the centroid of each cluster is recalculated based on these assignments and the data points are re-assigned based on these new centroids. This continues iteratively until the process converges to the clusters with the minimum within-cluster variance.
2. **Ward's clustering** method is a hierarchical agglomerative clustering method. Each data point begins as its own cluster, and nearby clusters are progressively merged. Ward's method merges the two clusters that lead to the minimum increase in within-cluster variance. Thus the distance between two clusters u and v, with u consisting of two subclusters s and t, is defined as

   with |x| being the number of data points in cluster x.

Results/Discussion Section -
We perform a variety of text analysis to analyze trends/patterns of Top 250 and Bottom 250 IMDb movies list.

K Means Clustering:
We experimented with k means clustering using several different values of k. However, we found that with values of k greater than 3 or 4 do not in general produce as good clusters with our data as they often have too much overlap and unclear topics. This is especially true of the bottom 250 movies, which tend to have a more limited vocabulary in their summaries. For instance, when clustering the bottom 250 movies with k=5 as seen in Figure 1 below, the orange cluster's common words will include "world," "become," "powerful," and "evil," and the pink cluster will include "evil," "world," "save," and "time," which is a significant overlap in topics.

Figure 1. k=5 clustering on Bottom 250 vs k=3 clustering on bottom 250

K Means: Top vs Bottom 250
IMDb's voters are predominantly male, and it is interesting to see how this shows in the ratings of movies. Common words found in the bottom 250 movies during preprocessing and clustering include "girls," "young," and "teenagers" while the top 250 movies more often focus on "man," "father," "son," or "wife." From this, it seems that bad movies may disproportionately focus on younger female characters, but it may be possible that IMDb voters simply more willing to criticize movies featuring this demographic. In comparison, the top 250 movies are male-centric, with summaries including "father" and "son" far more often than "mother" or "daughter" and referring to characters as "man" and "wife" rather than "woman" and "husband."
Running k means clustering on both the top 250 movies and the bottom 250 movies combined also gives interesting results. Figure 2 below visualizes the top 250 and bottom 250 movies using multidimensional scaling on the cosine distances from the tf-idf matrix. K means clustering with k=2 separates these categories with a 93% accuracy. The teal cluster, corresponding to the bottom 250 movies, commonly includes words like "evil," "world," and "powerful," as well as "young" and "girl," which we previously noted to be common to the bottom 250. The orange cluster, corresponding to the top 250 movies, include words like "life," "new," and "time," as well as "man" and "wife." Thus it seems that many "bad" movies may have simple good vs. evil plots while "good" movies are about more complex aspects of life.
The simplicity of bottom movie summaries shows up when looking at top and bottom movies separately as well. For instance, while bottom movie summaries might talk generally of "evil" and "killing," top movie summaries would use specific terms like "war," "gang," and "murder." The bottom movies overall have a much more limited vocabulary which reflects their limited plots.

Figure 2. Top 250 vs Bottom 250 movies on the left

Hierarchical Clustering: Top vs Bottom
Ward’s method clustering similarly has a 92.5% accuracy in differentiating top movies from bottom movies based on cosine distance between their summaries.
Figure 3 below shows a visualization of Ward’s method on the top 250 movies.

Figure 3. Ward’s method showing 3 clusters on the top 250 movies

LDA vs Clustering
Figure 4a shows the word clouds for 3 clusters and topics with k-means clustering and LDA topic modelling respectively. In general, LDA was able to find more specific and interesting small topics than clustering with higher numbers of topics, as can be seen by the example topics shown in figure 4b.

Figure 4a. Word clouds for 3 clusters/topics with k-means on the left and LDA on the right

Figure 4b. Word clouds for example clusters with LDA with 9 topics

LDA-

We use LDA, to understand the underlying topic structure of the documents. We also perform additional tests and analysis to confirm that the topic structure uncovered by LDA is a good structure.

Number of Topics = 8

Figure 5: Topic Representation with Most Important Words (Bottom Top 250 movies)

Selecting Number of Topics -

One of the problem with LDA is that several different values for k may be plausible, but by increasing k we sacrifice clarity. To combat this we use an elbow plot of Perplexity score v/s number of topics. As applied to LDA, for a given value of k, we estimate the LDA model. [7]

Then given the theoretical word distributions represented by the topics compare that to the actual topic mixtures, or distribution of words in the original documents. Perplexity is a statistical measure of how well a probability model predicts a sample.[10] However, the statistic is somewhat meaningless on its own. The benefit of this statistic comes in comparing perplexity across different models with varying ks. The model with the lowest perplexity is generally considered the “best”. For this evaluation, we iteratively generate a series of LDA models for the dataset, using a different number of topics in each model. We choose the *“elbow point”* as *Number of topics = 9* and show our results for the same.

Figure 6. Plot to determine “Optimal” number of topics

After evaluation on number of topics = 9, we are get more specific topics. We observe that the perplexity is still dropping at k=20 and keep on decreasing monotonically.

CMU Dataset:

We implemented a visualization of LDA topic modelling with the t-SNE algorithm, but it was very sparse since we only had 500 movies. We decided to use the the CMU Movie Summary Corpus for this visualization, which includes over 42,000 movie summaries.

While the CMU dataset does not include rating data, it does have year and revenue data. We experimented with using revenue data in the slider, but unfortunately not enough movies had revenue data to effectively visualize this. Since more movies’ metadata included year of release, we used this to add an extra dimension to the plot. Movies with no year data were marked as being published in 2019 so they can be easily removed from the plot by moving the slider one tick left.

We experimented with different numbers of topics and found that for this plot, topics greater than 5 had too much overlap and made for a messy visualization. Thus we decided to use 5 topics.

Figure 7. CMU Dataset with 5 topics and year data

Interpretation of Results -

1. **Top Ranked Movies**-
   a. LDA Modelling: We observe that the most dominant topic were described by - life, man, time, young, family, story. 85% of the tokens are captured by this topic. We interpret that most of the top ranked movies have the common theme of love and life as opposed to evil and killing.
   b. Clustering: The main three clusters found by k-means are war movies, murder movies, and life movies. Additional clusters found with higher k values include movies about gangs and families. The top-ranked movies tend to have pretty focused clusters, reflecting how top-ranked movies usually fit into specific themes and genres.
2. **Bottom Ranked Movies** -
   a. LDA Modelling: We observed that many of the bottom movies' topics represent the usual themes of "bad movies" - sharks, vampires, evil, porn/sex, and treasure all feature heavily. Terms like "college" and "sorority" are also common, which supports our previous observation that bottom-ranked movies often feature younger female characters. The bottom-ranked movies did not have a single dominant topic like the top-ranked movies, but instead was more split into several smaller topics. We interpreted this to mean that bad movies do not have a single common theme but instead are split into several possible themes.
   b. Clustering: The main three clusters found by k-means are movies about evil, girls, and trying/killed. These clusters do not seem to make as much sense as with the top-ranked movies with we interpret to mean that bottom-ranked movies do not have as focused of plots and themes.

Conclusion:
Our model provides a general framework for understanding the similarities that underlie movie's rating and their descriptions. The model can make cluster and generate topics to interpret the common themes prevalent in both the list. Furthermore, we have presented a number of interactive visualizations (website) which would allow an user to themselves analyze movie themes. In addition to this being of psychological interest, it is a useful framework for visualizations of other topic models.
Using LDA to find topics, our intuition and domain knowledge as a researcher is important. In our project we use perplexity as one data point in the decision process, but a lot of the time it might help to simply look at the topics themselves and the highest probability words associated with each one to determine if the structure would makes sense to the user.

Reference:
1. https://www.dbgroup.unimo.it/~po/pubs/LNBI_2015.pdf
2. http://journals.sagepub.com/doi/pdf/10.1177/1473871612439644
3. https://arxiv.org/pdf/1701.00199.pdf
4. Lin Liu, Lin Tang, Wen Dong , Shaowen Yao4 and Wei Zhou, "An overview of topic modeling and its current applications in bioinformatic, 2016"
5. Sonia Bergamaschi, Laura Po and Serena Sorrentino , "Comparing Topic Models for a Movie Recommendation System"
6.Topic Modeling and t-SNE Visualization, https://shuaiw.github.io/2016/12/22/topic-modeling-and-tsne-visualzation.html
7. "Text analysis: topic modeling", http://cfss.uchicago.edu/fall2016/text02.html#selecting_\(k\)

Team Contributions:
Jose: Data collection. Made the website. Helped with the report.
Dipali: Topic modelling, clustering, and visualization implementations. Helped with the report.
Bobby: Visualization collection and analysis. Made the presentation. Helped with the report.