Movie Plot Based Recommender System

Eyoel Armede, Andrew Cummings, Brad Fetes, Ian Ustanik

IST 664 Final Poster Presentation

Project Rationale

•Traditional recommender systems use a mix of ratings systems and clustering based on movies watched and similar movie genres or length.

•Our goal is to create an addition to these recommender systems by using NLP techniques that match users with relevant movies based on processing the plots of movies they have seen.

•Recommending movies based on the content of a film's plot will not only recommend movies that are similar to a user's likes and interests (what current recommenders do) but also expand their interest across genres and types of movies they may typically watch.

Data Description

•The datasets we are using for our analysis are from Kaggle.

•The information in these datasets come from one of two sources, IMDB or Wikipedia.

•Each dataset includes the title of the film, release year, plot summary, and a genre categorization.

•Two of the datasets add additional elements such as name of the director, prominent actors/actresses, Wikipedia page, and origin of the film.

•Combined, filtered dataset contained ~ 90,000 movie records.

•The team decided to only analyze movies that were released >= 1980.

Pre-Processing Steps

•The data is selected from 3 different resources as such the data set has been filtered and merged.

•All Features aren't available across all the 3 sources as such the necessary features were plot summary, title and release year .

•The datasets were combined in such a way that if the movie is available across all the data sets then the entry with the longest plot summary is taken as an entry.

•The data set was filtered based on origin as well only taking American and British movies.

	Release Year	Title	Origin/Ethnicity	Genre	Plot
15568	2009	(500) Days of Summer	American	romantic comedy	The film is presented in a nonlinear narrative...
14795	2006	45	American	crime drama	Big Al (Angus Macfadyen), and his girlfriend K...
15122	2007	10 MPH	American	documentary	10 MPH follows the progress of Caldwell as he ...
15381	2008	10,000 BC	American	adventure	At about 10,000 BC, a tribe of hunter-gatherer...
13592	2000	102 Dalmatians	American	comedy, family	After three years in prison, Cruella de Vil ha...

Feature Selection

•The features selected are the text of the plot summary only.

•We analyzed the sentiments of each sentence in the plot and created an average sentiment value of the movies based on the plot and used negative, positive, neutral, and compound sentiment values from VADER sentiment analyzer as part of our feature sets.

•The most common words were used as feature sets as well by removing stop words and names.

•The most common verbs, adverbs, adjectives were used as feature sets as well after removing stop words.

Model 1 (Most Common Words)

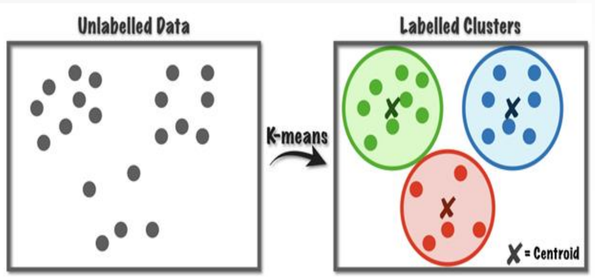
•One option for a recommendation engine based on movie plots would be to create features based on the most common words in the corpus.

•Our corpus contained over 60,000 words and this model utilized the top 2% of words in the corpus to look for relationships between movie plots.

•There are a few downsides to this potential model. First, by basing the features on the most popular words in the whole corpus, movies with unique plots may be left empty or without any defining features.

•Relatedly, picking the appropriate number of words to base features on is important. If too few word features are picked, distinctions between movies will be difficult but having too many words as features may render some clusters with only one or two films.

•Lastly, the difference in the lengths of the movie's plots is going to be a concern using this method. For example, Legally Blonde has 147 terms in which to cluster around, while the 1987 film Cyclone only has seven.



Model 2 (TF-IDF)

- Words common to all documents get discounted by the inverse of their frequency:

$$\text{tf-idf}_{ij} = f_{ij} \log \frac{|D|+1}{f_i+1}$$

  - fi is number of documents that contain word i

•Generated sentiment of each movie plot using NLTK VADER applied on each sentence, with the average of all the sentences returned.

•Used a PySpark machine learning pipeline:

1. Tokenizer()
2. StopWordsRemover()
3. CountVectorizer()
4. IDF()
5. VectorAssembler()
6. StandardScaler()
7. Kmeans()

•Produced K-means clusters, where k=100 distinct clusters.

•Few very large clusters where there was too little plot data and therefore insufficient distinction amongst individual movies.

•Few very small clusters that appeared to be outliers.

•Returned ~ 40 very strong candidates for good clusters which generated interesting insights and results that could be used to make a movie recommendation.

Title	Release Year	Genre	prediction
Lady Jane	1986	historical drama	98
My Sassy Girl	2008	comedy	98
One Night with th...	2006	drama	98
Restoration	1995	drama	98
Shrek the Third	2007	animation	98
Sword of the Valiant	1984	action	98
The King and I	1999	animated	98
Troy	2004	adventure	98

Title	Release Year	Genre	prediction
Bad News Bears	2005	comedy	10
G-Force	2009	family, live-acti...	10
Rat Race	2001	comedy	10
Zathura	2005	fantasy	10
Zero Charisma	2013	comedy	10
17 Again	2009	comedy, teen	10
Anacondas: The Hu...	2004	horror	10
Any Given Sunday	1999	drama, sports	10
Artie Lange's Bee...	2006	comedy	10
Battle of the Year	2013	dance, drama	10
Celtic Pride	1996	comedy	10
Church Ball	2006	comedy	10
D3: The Mighty Ducks	1996	family	10
Death Race	2008	action	10
Forever Strong	2008	drama	10
High School Music...	2008	musical, family	10
How to Make a Mon...	2001	science fiction	10
Internship, TheTh...	2013	comedy	10
Jumanji: Welcome ...	2017	action, fantasy, ...	10
Lake Placid	1999	horror comedy	10

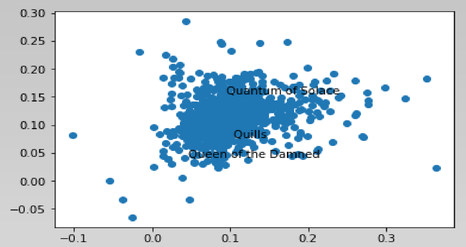
Model 3 (Recommender System

•This method uses word embedding and filtration to create vectors for each movie, a well-known algorithm used by companies such as Netflix.

•Recommender System using Words2Vector Pipeline:

1. Tokenizer()
2. StopWordRemover()
3. Word2Vec()
4. VectorAssembler()
5. Kmeans()

•This method is useful because of its high interpretability: features are in 3 dimensions, but also limited for clusters because the splits can be arbitrary.



Comparison

- Features: In model 1, the most common words were used as features, this contrasts our second model, the tf-idf model because tf-idf penalizes weights for most common words, while our third model has filtered words being embedded and compared directly without penalizing number of instances.
- Outcomes of each model: overall, each model contained 100 clusters, with all three models having one larger cluster and many smaller clusters that contained outlier movies.

Interpretation

•We wanted to utilize the plot to find movie recommendations that bypassed feature-heavy traditional genre, user review-based models.

•Our models achieved what we wanted to achieve, groups of non-traditional clusters that recommend movies that on the surface do not appear to be related.

•Our models also matched movies that one would expect to see together, showing their versatility and usability in a combined commercial recommender system.

Reference

1. https://www.kaggle.com/mananjhaveri/imdb-movies-data/tasks?taskId=2170
2. https://www.kaggle.com/jrobischon/wikipedia-movie-plots
3. https://www.kaggle.com/cryptexcode/mpst-movie-plot-synopses-with-tags