Capstone project - Report

Goals of the Project

Deciding what movie to watch with your significant other or with your friends can be the most difficult part of movie night! It is not only time consuming but a real source of struggle. Agreeing on a movie genre usually speed up the decision towards a common ground. Another way to help you decide which movie to pick is to check out the reviews to find out if a movie is worth watching. However, many times people come across spoilers that take away all the intrigue and ruin all the big moments!! As a movie lover, I have encountered these situations several times and that is what motivated me into designing an app that predicts movie genres and detects spoilers.

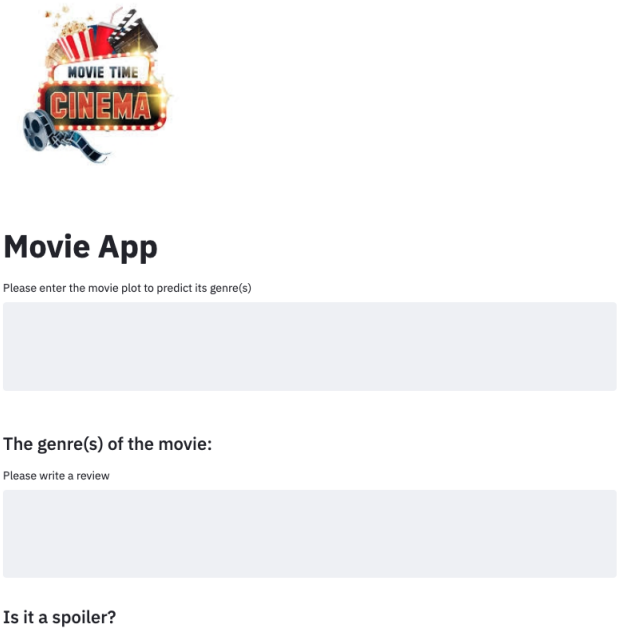

Figure 1: For the demo check the link https://www.loom.com/share/580d5eeb4d8b452c8f3807ea37b128b6.

Background

Predicting movie genres falls into the so called multilabel classification. In machine learning, this refers to a problem where multiple labels may be assigned to each instance. Unlike the traditional two-class and multiclass classification problems where classes are mutually exclusive, in the multilabel problem the classes are not mutually exclusive. Thus there is no constraint on how many of the classes the instance can be assigned to. To better understand the difference, please refer to fig. 2 There are different models that can handle the multilabel classification. The One-Vs-All model is one of them. As its name suggests, it is a strategy where you train multiple independent binary classifiers with one class at a time and leaving rest out. For instance, this can be translated to the following set of questions: Is this movie a Comedy or not? Is this movie a Drama or not? Is this movie a Thriller or not?... The main assumption here is that the labels are completely independent and we do not consider any correlation between the classes.

Determining if a review is a spoiler or not requires an analysis based on text similarity. The way to implement the latter is to use particular techniques to find the closeness between two chunks of text. Among the most popular statistical methods we can find Cosine similarity, Jaccard Similarity and Word Mover's Distance. All of them require preprocessing of the text before proceeding with the comparison.

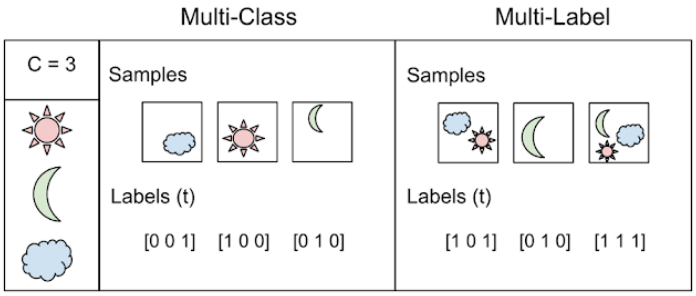

Figure 2: Difference between multicalss and multilabel classification

My Data

The first part of my project (predicting movie genres) required my dataset to have unique ids, movie genres and movie plots or synopsis. I soon realized that such a dataset does not exist and I that I had to scrape the IMDB website to get that. I ended up with a dataset with roughly 13500 movies that I collected over the last 70 years. The second part of my project (detecting spoilers in reviews) required to have a dataset that encapsulates movie ids, movie reviews and another column that specifies if a movie is a spoiler or not. I was lucky enough to find that a dataset that had this schema is already published in Kaggle (https://www.kaggle.com/rmisra/imdb-spoiler-dataset)

EDA and NLP

The distribution of the movie genres was highly unbalanced and could not get fixed due to the nature of the problem (multilabel classification) as shown in fig. 3. Roughly half my movies are tagged as Drama. In the other hand, I had movie genres with less than 300 counts. I decided to drop the genres 'FilmNoir', 'News', 'RealityTV' and 'Western' from the list of possible labels that can be assigned to a movie.

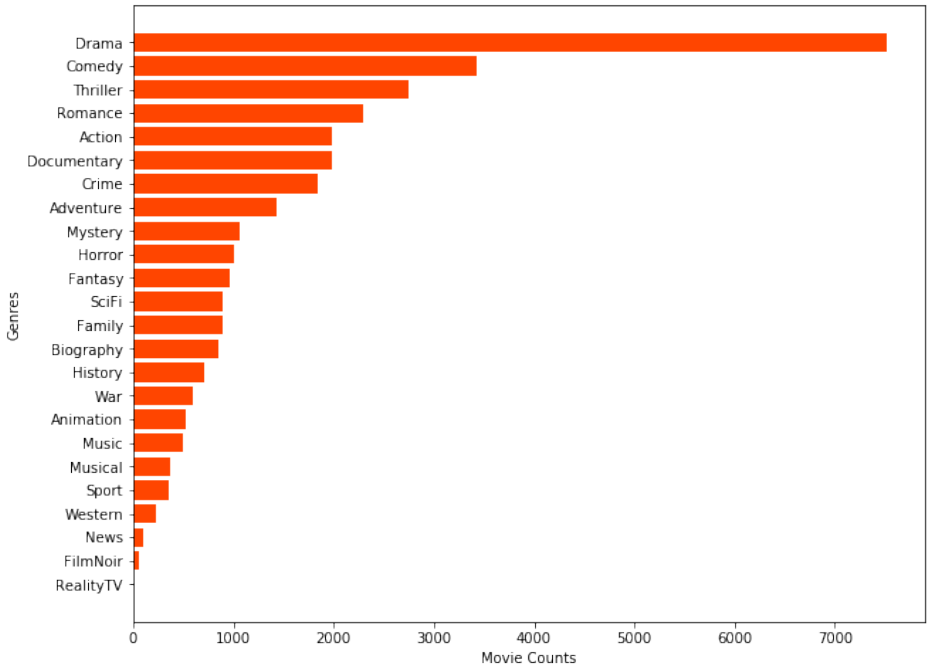

Figure 3: Huge unbalance of the genre distribution. Half of the movies are labeled as "Drama"

I preprocessed the text of the plots by writing a tokenizer that removed the punctuation, converted the sentences to lower case and removed stop words and numbers. The tokenizer also applied lemmatization to converts words into their initial root form. Stemming was also performed to cut off common word endings such as 's' and 'ing'. The tokenizer returns a list of tokens that goes as an input to another function called to_string. The latter returns a sentence made from the joined the tokens. The next step was to use CountVectorizer to transform the preprocessed text of the movie plots into encoded numerical values.

The number of reviews labeled as spoilers in my second dataset was much less than the number of the regular reviews. So I decided to downsample the dominant class. Silmilarly to my previous dataset, I performed the same preprocessing of the data which means that I used my_tokenizer and my to_string function to get cleaned text sentences. Unlike Jaccard Similarity and Word Mover's Distance, the Cosine Similarity method required that I use in addition NLP to transform the text into vectors. For the Word Mover's Distance, I modified my_tokenizer function to make sure not to include the lemmatizer and the stemmer because this method targets both semantic and syntactic approach to get similarity between text documents.

Modeling

I decided to use the One-Vs-All model along with Logistic Regression. I optimized the hyperparameters by conducting GridSearchCV. In order to do that, I defined a pipeline that scaled the data, fitted the different models to the data, and then tested several combinations of parameters to see which one will give me the best accuracy score. The selected model was scaled using Standard Scaler and had the following parameters (C= 0.01, penalty= 'l2')

For the text similarity part of my project, I also divided my data into train and test set and applied Cosine similarity, Jaccard Similarity and Word Mover's Distance to find which one will give me better results. For every single statistical method, I used my train set to optimize the threshold beyond which a review will be considered as a spoiler and tested it on my test set. I ended up choosing the Cosine similarity because it gave the best results and set up the threshold to be 0.0895.

Results

I performed an accuracy score on the overall test set which calculated the exact match of label combinations between the predictions and the true values. I got an accuracy of about 30%. This does not mean that the rest are not correct, it only means that the rest of the combinations were probably missing/misclassifying a labels amongst all the predicted ones. So I printed the accuracy scores of every single genres and I got a score of 92% on average for every single one. I also checked the confusion matrix along with the precision and recall for every genre. The conclusion id that, on average I have a good precision, the recall however changes drastically from one genre to another. This means that the model is performing thoroughly for some genres but not learning enough to perform well for some others. For instance, the model is struggling with recognizing Biography and History. It is however doing very well for Drama and Documentary. I also could extract the words that best describe a genre. I am providing this link to the Plotly interactive graph that displays that (https://plotly.com/ skrichen/1/).

The cosine similarity gave me 62% accuracy on the train set and 60% accuracy on the test set. I also checked the precision and recall that were also about more than average.

Findings, Conclusions and Future Directions

Overall my genre predictor is not performing that badly giving that much unbalance of the data. The only way to improve the model is to fix the skew of the distribution which is really difficult. I thought trying to reduce it by combining some labels together using the Market Basket Analysis. So, if News and War always show up together (in the same basket), it might be a good idea to merge the two labels as one. I also wanted to check some other models that won't treat every label as independent of the others but will take into account the correlation between the labels. One model that allows to do that is Label Powerset. However this approach needs wort case ($2^{|C|}$) classifiers, and has a high computational complexity. In other words, for a problem like mine with 20 labels, I will need to train roughly 1 million models to get my results. I honestly don't think it is worth it. So as future work, I should try to put more effort fixing the data unbalance and look at the results again.

For the text similarity part of my project, I was expecting better results but I would like to mention something though.. I checked the IMDB website for reviews with spoilers and I went through some of them and it turned out that several reviews were not disclosing any major info about the plots even though they were tagged as such. My guess it that the IMDB website classification is also based on a model that looks conservative and I was basically trying to fit my results to another model's results. If this is the case, scores and metrics won't really reflect the true performance of the statistical method.

That being said, many more sophisticated methods are out there ready to be used to determine text similarity such as GloVe and Siamese Manhattan LSTM and that I can check out for future improvements.

Business Applications

Multilabel classification with NLP can be used in different areas such as emails and news articles classification. Text similarity can be used in product recommanders.