Movie Genre Classification

Ali Amirfazli, Sarvesh Ramachandran, Willem Taylor

May 5, 2022

Abstract

In an effort to create a model to classify the genre of a movie based on its plot synopsis, a variety of approaches were employed including Multiple Naive Bayes, SVM with TF-IDF, and SGD with TF-IDF SMOTE and lemmatization. Each of these models showed approximately equivalent classification accuracy around 60%. They showed that while there are keywords prevalent among certain genres that can often indicate a movie's genre, many movie plots require a deeper level of reading comprehension due to high overlap in some genre categories and plot synopsis being left intentionally vague.

1 Introduction

The goal of this project was to build a multi-class classification model to classify the genre of a movie based on its plot description. An additional goal was to be able to identify which words gave the biggest indication of a movie's genre. The dataset contains information scraped from Wikipedia of 34,886 movies from around the world - including each movie's release year, title, country of origin, director, plot summary, and genre. The dataset was reduced to only include movies originating in primarily English-speaking countries to ensure all plots were written in English. To ensure the dataset had enough data on each genre category to adequately train and test, it was reduced to contain only movies from the top 9 most common genre categories: Drama, Comedy, Horror, Western, Thriller, Action, Adventure, Musical, and Science Fiction. The resulting dataset contained the plot summary and genre of 12,995 movies in English from the 9 most common genres.

The plot summaries were then parsed to make all words lowercase, remove punctuation, and remove numerical digits so common words across different plots could be recognized by the model. The python nltk stopwords library was used to remove common English stopwords from the plot summaries since these words do not provide any insight into the meaning of a text. Some further changes were made to the plot strings after some experimentation proved them useful, but these will be discussed later.

2 Data Exploration

With a working set of plot summaries and the key words they contained, our group did some initial analysis of the corpus to see what kind of patterns we could find in the plots. These patterns helped guide us in picking machine learning models that would perform accurately (See Appendix).

The results when viewing the top 50 most frequently used words are unsurprising, as we see words like "father", "home", and "tells" in the list, all vague and commonly found words in descriptions for all types of movies. However, this information gives us a baseline for usage of words overall. When we viewed how often words appear in a particular genre, (for example western movies), we started to see more interesting

results, as words we typically associate with the west, like “ranch”, “sheriff” and “gold” appeared.

Our experimentation showed promising results for genres having some kind of distinct glossary and that a bag of words approach for classification would perform well, since we could associate the occurrence of particular words appearing in a plot to a certain genre.

3 Experimentation

3.1 Naive Bayes

After this preliminary exploration showed patterns in the prevalence of certain keywords in certain movie genres, a Naive Bayes approach with a bag-of-words representation was chosen as this would allow classification based on the presence of certain words commonly used in certain genres. The manual Naive Bayes implementation with bag-of-words was similar to the one outlined in (Jurafsky, D., & Martin, J. H.) (see Appendix Figure 1).

The logprior and loglikelihood for each genre and each word were calculated based on the proportion of each genre in the dataset and the proportion of each word’s occurrence in each genre, and classification decisions were made based on which of the 9 possible genres had the highest log likelihood for a particular movie. The dataset was randomly split into a training set (70%) and testing set (30%) and multiple tests were run to assess the model’s predictive accuracy. The classification accuracy ranged from 60%-62%. A confusion matrix was generated which showed very high classification rates for dramas, comedies, westerns, and horrors. However, classification rates for the other 5 genres were extremely low. A vast majority of incorrectly classified movies were incorrectly predicted to be comedies or dramas, and an exploration of the dataset showed why this was taking place - there were drastically more drama and comedy movies in the dataset than any other genre. This strongly affected the logprior values of drama and comedy when training our Naive Bayes model, which made the model less likely to classify movies as genres other than drama or comedy since it is harder for the sum of loglikelihoods of words to overcome the initial priors. Though the model shows a relatively high predictive accuracy, it would not be effective when used to classify new data with a different (unknown) genre distribution, as it relies too much on prior knowledge about the genre distribution instead of focusing on the content of the plots.

To counteract this, the Naive Bayes model was instead trained and tested on a subset of the dataset that contained approximately equal numbers of movies from each genre by randomly sampling about 400 movies from every genre. After training and testing the model on the dataset containing approximately equal numbers of each genre, the classification accuracy still sat around 60-61%, but a much greater portion of movies outside of comedy and drama were classified correctly. The confusion matrix for one experimental run can be found in Appendix Figure 2, and it showed a drastic improvement in classification accuracy for every genre except comedy and drama.

As mentioned earlier in the report, a secondary goal was to understand which words were most influential in assigning a movie to a particular genre, as it was believed that insight into this could help provide a better understanding of which words should not be considered in the dataset. The key to discovering which words contributed most to a particular movie being classified as a certain genre is determining which words from the document increased the sum of log likelihoods for the predicted genre by the most relative to the other possible genres. This can be represented as the difference between a word’s loglikelihood for the chosen genre and the mean of the word’s loglikelihood for all other genres:

Many of the most common influential words for each genre matched what a human

would commonly associate with each genre, but some proper nouns such as names were also included in the list. These should not hold much weight in determining a movie's genre, and including names of actors could give the model additional unfair information. To counteract this, a step was added in the preprocessing of the dataset to remove any capitalized words that occur outside the beginning of a sentence to remove any proper nouns influencing classification. The updated list of most important words for the different genres makes the most sense compared to what a human would expect to see associated with each genre, and it can be seen in Appendix Figure 3. Running Naive Bayes on the newly modified dataset did not reveal any significant improvement, though - some genres classification accuracy improved very slightly, but others went slightly down, and the model's overall accuracy remained at 61

Experimentation with Naive Bayes showed that the prevalence of certain keywords shows great promise in predicting a movie's genre. However, it also revealed that action would need to be taken to counteract the unequal distribution of genres in the dataset, and indicated a need for further investigation into the words impacting genre classification outside of those most common words discussed above.

3.2 TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. This is a data preprocessing step that would replace bag of words. Rather than focusing on how often a word would appear within a particular corpus, TF-IDF tries to add an additional emphasis on words that do not show up in many documents within a corpus.

$$TFIDF = \frac{\text{frequency of word in paragraph}}{\text{length of paragraph}} * log(\frac{\text{\# paragraphs in corpus}}{\text{\# paragraphs word occurs in}})$$

Given this calculation, we would expect words that do not appear often to be weighted higher. Although we have already removed stop words from the word bank, this method still has value because we can lower the weights on the vague words that all genres seem to share and make the model focus more on the more distinct words in each synopsis.

One particular advantage of using TF-IDF is that it allows us to fit a model such as SVM, while maintaining the log penalty in frequent words. However, trying to fit SVM with a TF IDF matrix did not particularly improve our results. Even with testing multiple kernels, the best accuracy we were able to obtain was around 55%. One reason may be that TF IDF is penalizing words that may be occurring often in only one genre. This limitation is what led us to trying a more 'class-based' approach to TF IDF. This changed the preprocessing equation to being based off frequency, length, and number of genres rather than paragraphs

This approach was not very effective for this set of data because it ultimately ended up penalizing the words which we considered 'important'. Although a word like 'ranch' had a huge proportion of its occurrences in western movies, it only needed to show up once in another genre's synopsis to lose overall weight. This ended up happening too often, and our feature matrix lost too much information. However, our group was still concerned that keeping common words in our feature matrix was confusing the model. Thus, the final modification we made to TF IDF was to remove words that occurred over a certain threshold. When using sklearn, this is the max df argument that is passed to the TFIDFVectorizer. Theoretically, this would strike the right balance between the vanilla TFIDF preprocessing, and our 'class based' approach. Common words would be penalized, but the penalization would not be so strong that even unique words would see that penalization. Through experimentation, our accuracy rose to an average of 63% for a linear SVM model fitted on a TFIDF feature matrix with max df set to 0.2.

3.3 Black Box Implementation

We next compared the results of the Naive Bayes and TFIDF SVM with a black box Natural Language Processing implementation using a five step data pipeline. The first two steps were intertwined: count vectorization and lemmatizing. Count vectorization took all unique words in all documents and counted the frequency of each word for each document. The lemmatization tokenizer uses dictionary definitions to determine connections between word stems, so it combines all similar words into a single token since multiple of the most important words detected by the Naive Bayes approach shared similar roots (such as "human" vs. "humans" in science fiction).The third step of the pipeline was a TF-IDF function, similar to that explained in previous sections. The final pre-processing step was SMOTE (Synthetic Minority Over-Sampling Technique). Since our dataset heavily overrepresented comedy and drama, SMOTE helps balance the dataset by using K-Nearest Neighbors to find an average of data points in one label. In the gap between the location of the average and the location of the data points, it randomly created synthetic data points to pad out the smallest genres of the dataset.

With these processing steps completed, we ran the information into a Stochastic Gradient Descent (SGD) classifier. We implemented two loss functions for the classifier: the main being a 'square hinge' loss, which worked similar to a linear SVM function with a quadratic penalty, and a 'log' loss, which worked similar to a logistic classifier. We then needed to choose and fit our models by hypertuning model parameters to the data. We manipulated 3 variables for the count vectorization: max df (which disregards words with a document frequency over the max df, min df (disregards words with a document frequency below the min df), and the n-grams (linear combinations of words - either single, two consecutive, or three consecutive words). For the SGDClassifier, we manipulated the regularization factor of the function: alpha.

The models were fit and hyperparameters tuned using three different datasets - the full trimmed dataset, passed through the pipeline before any manual processing of plots; the trimmed dataset with manual processing of plots, passed through the pipeline; and the trimmed dataset with manual processing of plots and no pipeline. The results once the model parameters were selected and the models were fit can be found in the appendix. In general, the first dataset did the best with 66% accuracy. However, all three cases performed similarly - in the 60-66% accuracy range, which is similar to the previous implementations.

4 Conclusion

The accuracy that we could expect our classification model to obtain was severely limited simply due to the nature of the problem. Even a human reading the plot summaries of a movie is not always able to discern a movie's genre. Further, there can be a lot of ambiguity in the genre of a movie - vague categories such as "comedy" can involve a wide variety of plots in a wide variety of settings, some of which could be satirical representations of other existing movies/genres, and it can be hard to nail down common plot elements of a comedy movie. Many movies also can be combinations of 2 or 3 different genres of movie, but this dataset forced most of them to be marked as one specific genre. Ultimately, in our dataset containing 9 possible genres, we would expect an 11% chance of genre classification based on random chance. Building a model that is able to obtain 60% classification accuracy without utilizing deep learning is a significant improvement, and this model showed a lot of insight into which words are most important in making a genre prediction while also displaying the limitation of a model's ability to pick up patterns that may not always be there.

Appendix

function TRAIN NAIVE BAYES(D, C) **returns** log $P(c)$ and log $P(w|c)$

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d $\in$ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) **returns** best c

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Figure 4.2 The naive Bayes algorithm, using add-1 smoothing. To use add-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

Figure 1: Naive Bayes Pseudocode

		Predicted Class								
		comedy	adventure	drama	western	thriller	horror	musical	action	science fiction
Actual Class	comedy	52	6	15	0	2	5	15	9	2
	adventure	3	69	6	4	0	7	0	11	11
	drama	17	8	42	1	24	9	5	13	8
	western	4	9	1	89	1	1	0	6	0
	thriller	6	3	4	1	48	31	1	13	14
	horror	0	2	0	1	3	84	0	0	10
	musical	18	4	10	3	3	4	78	3	1
	action	3	9	0	1	11	2	2	68	10
	science fiction	1	0	2	0	3	12	0	5	89

Figure 2: Confusion Matrix

```
drama:  ['mother', 'marriage', 'sexual', 'love', 'affair', 'marry', 'grandmother', 'crying', 'marrying', 'sex']
comedy:  ['wedding', 'win', 'bet', 'advertising', 'marry', 'coach', 'fail', 'wins', 'senior', 'job']
horror:  ['corpse', 'blood', 'creature', 'strange', 'basement', 'body', 'throat', 'stabs', 'vampire', 'spirit']
thriller:  ['police', 'apartment', 'killer', 'murder', 'affair', 'phone', 'case', 'agents', 'gunpoint', 'innocence']
western:  ['ranch', 'rides', 'saloon', 'outlaw', 'rancher', 'posse', 'cattle', 'marshal', 'outlaws', 'stagecoach']
action:  ['bomb', 'team', 'truck', 'drug', 'agents', 'hostage', 'training', 'helicopter', 'shootout', 'henchmen']
musical:  ['singer', 'band', 'show', 'music', 'musical', 'song', 'dancer', 'star', 'producer', 'love']
science fiction:  ['humans', 'human', 'space', 'planet', 'alien', 'creature', 'ship', 'giant', 'scientists', 'spacesh
ip']
```

Figure 3: Important Words by Genre

References

A. C. Saputra, A. B. Sitepu, Stanley, P. W. P. Yohanes Sigit, P. G. Sarto Aji Tetuko and G. C. Nugroho, "The Classification of the Movie Genre based on Synopsis of the Indonesian Film," 2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT), 2019, pp. 201-204, doi: 10.1109/ICAIIT.2019.8834606.

Jurafsky, D., Martin, J. H. (2021, December 21). Speech and Language Processing (3rd ed. draft) - Naive Bayes and Sentiment Classification. Speech and Language Processing. Retrieved May 2, 2022, from https://web.stanford.edu/ jurafsky/slp3/4.pdf