Classifying Movie Genres Based on Plot Summaries

T. Huang, I. Borzov, D. Cucuzza

CLASSIFYING MOVIE GENRES BASED ON PLOT SUMMARIES

Thomas Huang, **Illia Borzov**, and **Daniel J. Cucuzza**

Abstract

Genre classification is a fundamental application that arises when ideas or contextual information are constructed using any type of language. Alongside the ever-growing expansion of the world wide web and the information it houses, automated classification serves the valuable purpose of grouping similar bodies of information together for organization and pattern detection. In this paper, we focus on the task of classifying movie genres based on their summaries.

1 Dataset

Our system is trained on the Carnegie Mellon University Movie Summary Corpus that contains metadata regarding roughly 42,000 movies extracted from databases such as Wikipedia. In total, there are 363 unique genre tags provided in the data set initially. Each entry in the set contains metadata for information such as the movie's name, release date, genre(s), and the most important feature for our system: plot summaries. Using this as our corpus, we create a table of movies and associate the list of classified genres and its plot summary with each movie entry. For clarification, project members worked in a shared Jupyter notebook in Google Colaboratory.

2 Preprocessing

In order to make use of the provided data set, the data collected must be cleaned and munged in order to be fed to our system. There are several preprocessing techniques that we have taken in order to improve our system's ability to parse the data. Firstly, we start off this discussion with token normalization. This means that every word located in each movie's plot summary will be set to lowercase and stripped of special characters in order to merge forms of words that share the same meaning semantically. For example, tokens like "Apple" and "apple", as well as "ap-ple" and "apple!", will share a normalized form and be treated identically. After this step, we remove words in each plot summary that carry little to no information about what is contextually being discussed (common examples are articles like "the" or some personal pronouns like "she"). This is known as stopword removal and is important for "trimming the fat" of each bag of words associated with a particular movie. Finally, we employed NLTK to lemmatize each tokenized word in the movie's plot summary. A more time-intensive but generally more robust way of token grouping than stemming, lemmatization uses a dictionary to convert each token to its base dictionary form; for example, words like "am", "are" and "is" become "be". After preprocessing, the top 50 most commonly occurring tokens are shown below.

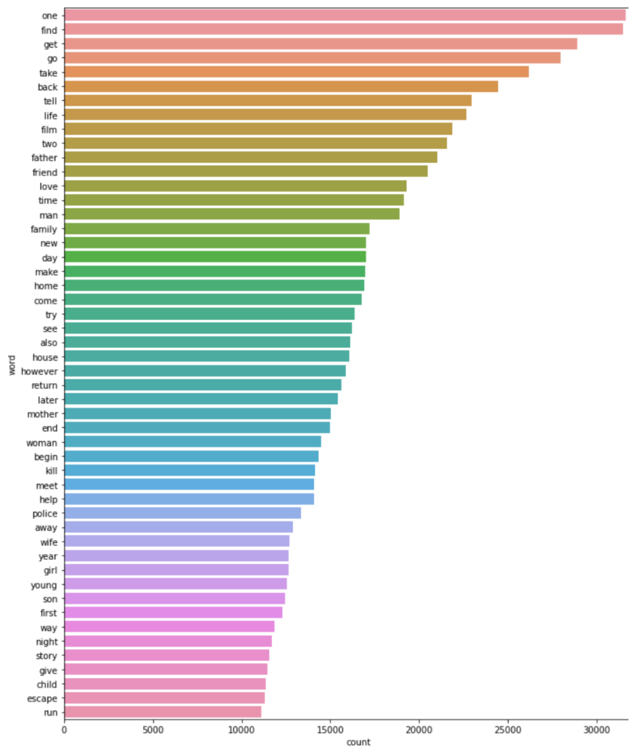

Figure 1 - Top 50 Occurring Words

3 Initial Experiment

Making use of the munging and cleaning techniques discussed prior, our initial pipeline was rather barebones. Our initial fundamental problem concerned generating vectors out of our now-tokenized summaries that we could feed into a machine learning model. Out of the options we explored, our first attempt used the Word2Vec algorithm to generate an n-dimensional vector for each word in a pool of words (in our case, this pool is the set of tokens in all summaries from our corpus). Word2Vec is a neural network model that generates word embeddings in the form of vectors. Similarities in two vectors represent associations the model has detected between those two corresponding words; for example, "death" and "kill" have similar vectors. We used the skip-gram variation of the model (which offers some contextual-meaning advantages over its alternative, continuous bag of words). Although the default for Word2Vec uses vectors of size 300, we opted for 100 to account for the relatively small size of our corpus (relative to other tasks in the embedding space) and ease of retraining in development.

However, despite these vectors' effectiveness in reflecting similarities between words, we still had to use these word vectors to generate a singular vector for each summary. To this end, we first tried using the average of each summary's word vectors. When fed through a generic random forest regressor, this resulted in a rather poorly-performing system with an f-score of about 0.16. A major possible flaw of this method is its failure to take into account the varying importance of each word in contributing to the overall meaning of its containing summary. We believe that weighing each word vector by some measure of its significance, like its TF-IDF score, would have yielded somewhat better results than what was achieved.

4 Improvements Made

We pursued several avenues of change in our efforts to refine the model. Instead of implementing TF-IDF, we opted to transition from Word2Vec to Doc2Vec, a pre-built neural-network generalization of the Word2Vec model that instead generates n-dimensional vectors for each document in a collection of documents. This vectorized system once again captures the similarities and differences between each document (in our case, summary) in the provided collection. Using our pre-processed tokens and their respective summaries, we trained a Doc2Vec model on a training set of about 30,000 summaries, capable of taking any string of words as input and generating a vector within the space of our training context. In the interest of time and reducing noise, we opted to exclude any words that appeared fewer than 15 times in our model's training, as it's unlikely they appear in enough contexts to build a strong vectoral context. After some testing, we upped the vector size back to the standard of 300, as improvements in accuracy of higher vector sizes seemed to show diminishing returns.

Another significant improvement addressed the 363 unique genres provided by the original corpus. While some degree of granularity is useful and necessary in classification, a quick visualization of the distribution of genres revealed that many were either niche or very sparsely represented.

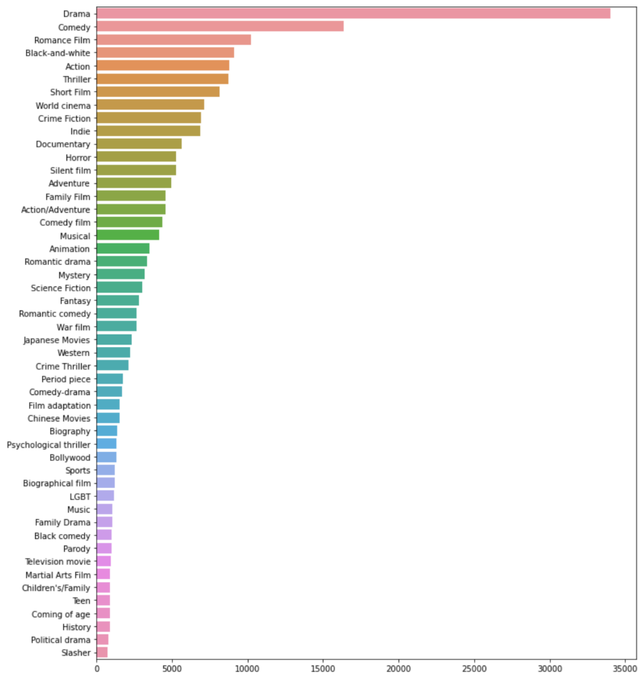

Figure 2 - Top 40 Genres (of 363) Before Grouping

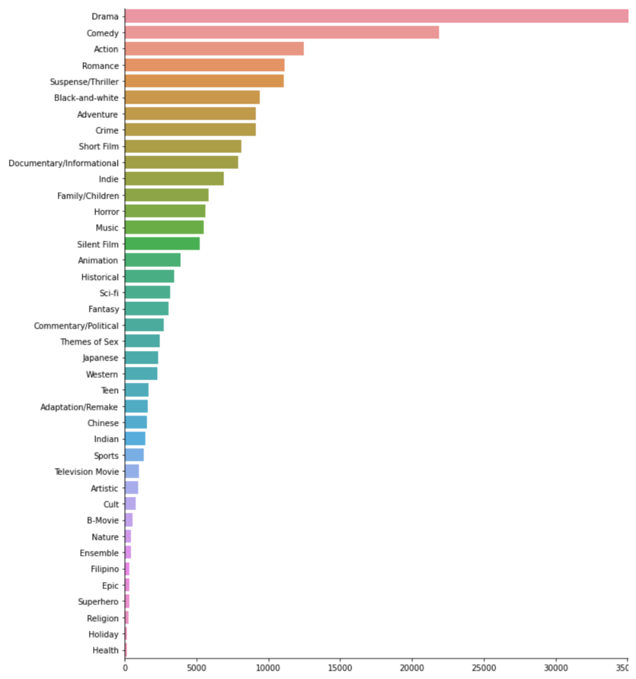

Figure 3 - Categories After Grouping

We can see in the figure above that a very large proportion of our movies were tagged as "Drama", whereas some tags like "Silhouette animation" had as few as a single movie. The sheer quantity of relatively trivial target genres was at least a time-sink for our training, and at worst a source of noise for our model. To address this, we condensed the 363 genres given down to 40 broader but still representative categories. This was initially done manually at our own discretion, but we later took into account significant correlations between genres in our clustering decisions.

It should be noted that our Doc2Vec model was trained on our training data, but the actual classification and system modifications made accordingly were built based on our development set. Having generated vectors for each summary using our Doc2Vec model, as well as having one-hot encoded genre target vectors for each summary, we fed this training data into a one-vs-rest classifier, using simple logistic regression to determine the likelihood of each genre individually for a summary, and we achieved an f-score of roughly 0.503; after loosening the criteria for classification by lowering the threshold from 0.5 to 0.25, we were able to improve our score to 0.544.

We then generated confusion matrices for each genre; as expected, common tags like "Drama" and "Comedy" experienced a high proportion of false positives due to their much larger sample sizes and variation in summaries. These confusion matrices were also used as reference for further modifying our genre grouping, with the goal of reducing false positives/negatives in development that would ideally generate less confusion in eventual training/testing. It should be noted that our Doc2Vec model was trained on our training data, but the actual classification and system modifications made accordingly were built based on our development set.

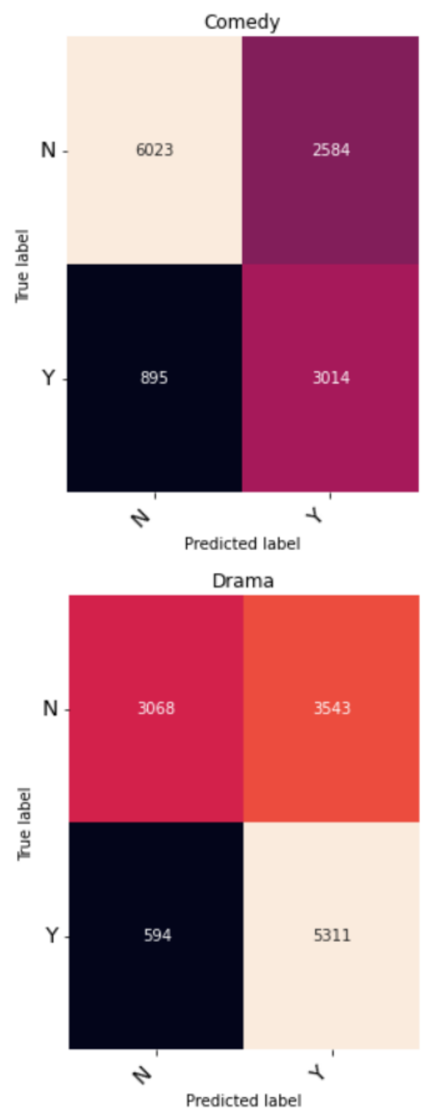

Figure 4 - Confusion Matrix Examples

5 Diminishing Returns

With the above improvements in place, we further attempted to strengthen our classification system by researching other multi-label classification models. The inspiration for this decision came from the fact that we had previously used a random forest classifier in our system that performed very poorly (see Section 3) when we naively took the average of a plot summary's feature vectors. Taking this previous attempt into account, we opted to properly experiment with other alternative classifiers in hopes of seeing even more improvements to our system. After integrating Doc2Vec into our model, we wanted to compare Scikit Learn's random forest classifier against the original logistic regression model. Initial expectations were that it would perform better than the currently engineered implementation. However, we quickly learned that this classification model did not even come close to competing with our logistic regression model, in large part because movies were either tagged with "Drama" or with nothing at all.

After attempting in vain to improve the results, we concluded tentatively that our dataset's skewed distribution was the culprit for this classifier's inefficiency. In essence, the distribution of already classified plot summaries tended heavily towards "Drama" (see Figure 1); in addition, Drama is a broad genre that can exist alongside many other genres that have few similarities within themselves (for example, horror and comedy), resulting in our random forest learning a connection between Drama and a variety of summary types. Though we could technically reduce the number of categories further in an attempt to even out the distribution, we believe that few, if any, of the 40 categories we currently have can be merged with each other without losing an important degree of genre-defining specificity; in this sense, we don't think that jamming sparsely represented but distinct genres together is an intelligent solution to a problem whose roots lie in the bias in distribution of our particular corpus. Though this conclusion is generally dissatisfactory, it's worth noting that random forests as a model are relatively prone to these issues, as decision trees are particularly sensitive to imbalances in feature distribution. Unsurprisingly, the random forest classifier system's outputted f-scores ranged between 0.19 and 0.25.

As an attempt to rectify this issue, we turned to a modified random forest classifier that better deals with unbalanced datasets as input (Balanced Random Forest Classifier). The particular techniques used to circumvent unbalanced data in this classification system are out of the scope of this discussion; however, the balanced system's f-score was significantly improved to a range between 0.39 and 0.47 after some additional parameter tuning. On a side note, our group additionally experimented with a one-vs-rest classifier using a support vector machine model. While the SVC outperformed the balanced random forest model with f-scores ranging from 0.47 to 0.51, it still failed to outperform logistic regression.

5 Conclusions

In summary, we used data from the Carnegie Mellon University Movie Summary Corpus of 42,000 movies and summaries, converted summaries to vectors using Doc2Vec, condensed the 363 given genres down to 40, and experimented with different classification models, the best-performing of which turned out to be logistic regression with an f-score of 0.544. In similar projects we encountered, f-scores rarely broke 0.6, and it became increasingly apparent as we worked that our dataset's distribution was not particularly conducive to multi-label classification, so in the end we are reasonably satisfied with our results. It has become apparent throughout our work that genres themselves and their relationships to movie summaries are nebulous and heavily dependent on subjective decisions made by those who label genres or write summaries; while we were able to capture a moderate relationship between them in our data, there is a lot of room to improve. Our system could likely be improved in several ways, including better grouping of genres, alternate model selection, experimenting with more nuanced/partial-credit scoring strategies, or even switching to a more modern or better distributed corpus.

6 References

1. Varshit Battu, Vishal Batchu, Rama Rohit Reddy Gangula, Mohana Murali Krishna Reddy Dakannagari, and Radhika Mamidi. 2018. Predicting the Genre and Rating of a Movie Based on its Synopsis. In Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation, Hong Kong. Association for Computational Linguistics. https://aclanthology.org/Y18-1007/

2. A. M. Ertugrul and P. Karagoz, "Movie Genre Classification from Plot Summaries Using Bidirectional LSTM," 2018 IEEE 12th International Conference on Semantic Computing (ICSC), 2018, pp. 248-251, doi: 10.1109/ICSC.2018.00043. https://doi.org/10.1109/ICSC.2018.00043

3. Ning Fei and Yangyang Zhang. 2019. Movie genre classification using TF-IDF and SVM. In Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City (ICIT 2019). Association for Computing Machinery, New York, NY, USA, 131–136. https://doi.org/10.1145/3377170.3377234

4. Marina Santini, Richard Power, and Roger Evans. 2006. Implementing a Characterization of Genre for Automatic Genre Identification of Web Pages. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 699–706, Sydney, Australia. Association for Computational Linguistics. https://aclanthology.org/P06-2090/

5. Joseph Worsham and Jugal Kalita. 2018. Genre Identification and the Compositional Effect of Genre in Literature. In Proceedings of the 27th International Conference on Computational Linguistics, pages 1963–1973, Santa Fe, New Mexico, USA. Association for Computational Linguistics. https://aclanthology.org/C18-1167/