Movie script genre classification

Konstantinos Mavromatakis
Gothenburg University
gusmavko@student.gu.se

Abstract

Movie genre classification is a quite popular, yet challenging NLP task. In this project, we investigate how informative movie scripts are with regards to genre. We test and compare different word representations as features and different classification algorithms and look into which method is the most successful in predicting a correct class label.

1 Introduction

Movie genre classification is a really popular classification task that has been researched to a great extent. Both unimodal and multimodal approaches, which utilize a variety of textual, visual and audio features, have been explored recently. Chu and Guo (2017) use film posters and, by focusing on what appears on the image, they train a Convolutional Neural Network for film genre classification. Ertugrul and KARAGOZ (2018) claim there is a hidden representation of genre information in the movie plot summaries. They estimate a genre for each sentence of a movie summary separately and, then, use majority voting for the final decision by considering the posterior probabilities of genres assigned to each sentence. Zhou et al. (2010) divide trailers into shot units and classify each scene into a genre using shot analysis. Lastly, Arevalo et al. (2017) perform movie classification by combining poster and plot information and argue that the multimodality of the data seemed to improve the results of their model.

Even though there is a significant amount of research in genre classification of certain types of text, like movie reviews and summaries, movie scripts have not been researched to the same extent. (Blackstock and Spitz, 2008) split the text into dialogue and stage direction parts and extract various NLP features from both parts. Then, they compare the performance of a Maximum Entropy Markov Model and Naive Bayes Classifier in the classification. Nakano et al. (2019) combine screenplay content and structure to train a Support Vector Machine (SVM) to predict genre labels.

The proposed paper builds on previous research by utilising different Machine Learning algorithms (ML), along with both simpler (e.g.TF-IDF) and more complex (e.g. BERT embeddings) meaning representations as features in the task of movie script classification. The classification happens purely on the basis of the script itself (i.e. we do not use information about movie script structure), to find out if classifiers can consistently predict movie genres based on such a peculiar type of text.

In Section 2, we describe the materials and methods used in this movie genre classification task. Section 3 presents the results yielded by the different algorithms, which are then discussed and analysed. Finally, Section 4 draws the conclusions and ideas for future work.

2 Materials and methods

2.1 Materials

2.1.1 Movie Scripts Dataset

The Movie Script Dataset[1] includes approximately 2000 movie titles, which are labelled with a unique genre and are demonstrated in Figure 1. The labels in the original dataset were integers, which were not meaningful. Thus, we manually annotated a random sample of 10 movies from each integer class with a more descriptive label. Based on the majority of the labels, the integer class was replaced. As seen in Figure 1, there is a significant class imbalance, which can potentially skew the results, but in the context of this project, we decided to not oversample.

[1] https://analyticsindiamag.com/movie-script-classification-hackathon/

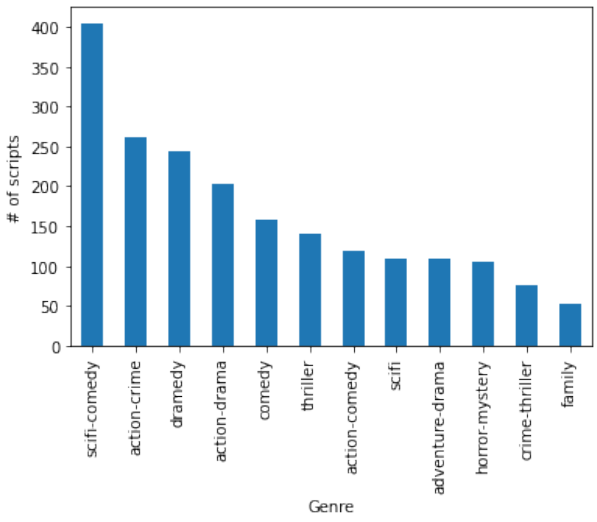

Figure 1: Movie genre distribution in the Dataset

2.2 Methods

2.2.1 Classification with Non-Contextual Features

Regarding the cleaning up part of the scripts, punctuation and digits were removed since they do not contribute substantial semantic information in the context of movie genre classification. After tokenizing with the RegExpTokenizer, the text was lowercased and lemmatized to limit the size of the vocabulary. Stopwords were removed from the text since they also do not seem to provide any semantic meaning.

Following the cleaning up part of the textual data, we chose a train-test split of 20% since it has proved to be a fair split in ML tasks. Then, we used TF-IDF to create vector representations of the texts, based on the Bag of words (BoW) model. A threshold of 10000 features was set, to prevent the vectorization of some exceptionally rare words. TF-IDF was chosen since it performs better than the BoW model by taking into consideration the importance of the word in a document.

Afterwards, four different classification algorithms were employed and tested for accuracy:

- Naive Bayes Multinomial Classifier,
- Support Vector Machine (SVM) classifier,
- KNeighbors Classifier,
- Logistic Regression

We calculated accuracy and F1-score to evaluate the performance of the classifiers that used TF-IDF features.

2.2.2 Classification with Contextual Features

More complex word representations were also implemented to investigate if they would be able to outperform the simpler ML algorithms that were trained on TF-IDF features. For this task, we used a distilled, uncased model, DistilBERT, that has about half the total number of parameters of BERT base, runs 60% faster and preserves approximately 95% of BERT's performances on the GLUE benchmark (Sanh et al., 2020). As Hinton et al., 2015, 1 describe, distillation is "a compression technique in which a small model is trained to reproduce the behavior of a larger model (or an ensemble of models)". The distilled version of BERT was chosen to avoid long times of training and to limit the envrionmental impact of the model.

BERT models use a special, subword-based tokenization method, called WordPiece tokenization (Devlin et al., 2019). It segments the input text via a longest-match-first tokenization strategy, known as Maximum Matching or MaxMatch. This approach aims to achieve a balance between vocabulary size and out-of-vocab words. For instance, "strawberries" would be split into "straw" and "berries" by the BERT tokenizer.

To avoid memory problems, we took a total of three steps. We processed one movie script at a time, instead of processing a batch of scripts. After each script was tokenized and vectorized, we deleted it. On top of that, because of the large size of the movie scripts (with an average of 83244 tokens per script), transformers models are not able to process sequences longer than 512 tokens at a time, truncating anything beyond this length. To mend this limitation, we break each script into chunks containing a maximum of 512 tokens each. After vectorizing and averaging the textual chunks, we apply a Logistic Regression algorithm on them, which yields movie genre labels for each script.

To obtain a better understanding of the model's performance, we applied Stratified K-fold cross-validation. Cross-validation is a resampling procedure used to evaluate ML models on a limited data sample. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not seen during the training of the model. In our task, we do stratified cross-validation, meaning that the same proportion of different classes will be included in each of the 5 folds we decided to test on.

Classifier	Accuracy	F1-Score
Naive Bayes	**18.69%**	**31.49%**
SVM	13.89%	19.76%
KNN	10.61%	12.82%
Logistic Regression	14.39%	21.07%

Table 1: Classifier performance when using TF-IDF as features

3 Results and Discussion

As described in Section 2, we implemented and compared two approaches with regards to the task in question: one that was not context-sensitive and one that was context-sensitive. The results of the first approach, which involved the use of TF-IDF as features and a variety of ML classification algorithms, are shown in Table 1.

We can observe that there are some discrepancies in the performance of the four algorithms. The Naive Bayes classifier and the Logistic Regression algorithm significantly outperform the SVM and KNN classifiers, with regards to both accuracy and F1-score. In general, Naive Bayes is more appropriate for solving multi-class prediction problems and also requires much less training data. Thus, in our case, where we have a multi-class task and a limited amount of data, Naive Bayes expectedly performed better. Expectedly, SVM outperforms KNN since the TF-IDF features used were large and the training data was limited. While the performance of the Naive Bayes classifier is higher than luck, it is still not equivalent to the performances of classifiers in tasks, such as movie summary classification (Arevalo et al., 2017).

A possible reason for the poor performance of these ML classifiers could be the type of features we extracted from the movie scripts. It is apparent that TF-IDF has several limitations. As TF-IDF is based on the BoW model, it does not capture position in text or word co-occurrences across different documents. For instance, the semantically ambiguous word "kill", which is likely to appear in both horror movies in its literal sense, and in comedies in its figurative sense, would be treated in the same manner by the algorithm. Therefore, TF-IDF would not be able to capture uses of words across varied contexts, which could then be employed by a classifier to make a distinction between two different movie genres. On top of that, it assumes that the counts of different words provide independent evidence of similarity. Finally, TF-IDF does not take contextual information into consideration, in the sense that it makes no use of semantic similarities between words.

Given that the highest accuracy and F1-score achieved by the Naive Bayes classifier are not above the 20% mark, we decided to employ more complex word representations as features to investigate if more complexity in the features would yield more promising results.

Across the 5 folds, the average accuracy score yielded when using contextual embeddings from the distilled BERT model is higher by approximately 7% in comparison with the Naive Bayes classifier. The accuracy change among the 5 folds can be seen in Figure 2. Expectedly, the accuracy on the training set is quite high while the accuracy on the test set tends to be lower. Testing our algorithm on five different test sets shows that there is a consistency in our algorithm's performance and our results are not merely because of chance.

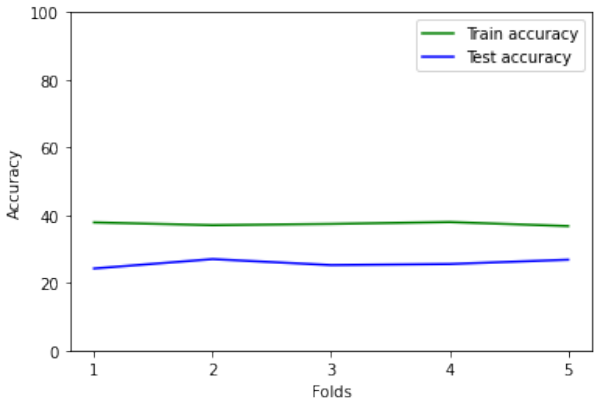

Figure 2: Train and Test accuracy across 5 different folds

To take the evaluation of the distilled model a step further, we decided to investigate the performances of specific genres. Certain genres' metrics seem to convey quite interesting information. For instance, scifi-comedy is the film genre with the lowest accuracy (18.21%) and F1 score (14.23%). It is likely that the low accuracy is related with the large number of false positives and false negatives in the prediction. Since it is the most frequent film genre in the dataset, the model seems to have a bias towards assigning the scifi-comedy label to multiple movie scripts. The model that used the BERT embeddings as features had the over-tendency of assigning the scifi-comedy label to the movie scripts.

After looking into the predictions of the action-crime movie genre, which is the second most fre-

quent genre in the dataset, it also appears to be assigned quite frequently to movies, yielding a greater number of false positives, similarly to the scifi-comedy genre. This can be explained because of the existence of bias towards the action-crime genre in the data.

Looking into the performance of the Naive Bayes classifier also shows the same over-tendency of assigning the most frequent movie genres to most movies.

In general, while more research is required to be conducted on the task in hand, this initial experiment and its results lead us to raise doubt about a system's ability in the classification of movie scripts. It appears that more complex meaning representations were not sufficient to boost the accuracy score of the classifiers by a large margin. It is likely that the small improvement in the performance may be due to the peculiar nature of our data: looking at the scripts in our dataset shows that they are often not structured in the same way, and simply using linguistic features as input to the model is not enough to provide a satisfying performance.

4 Conclusion

In this paper, we compared word representations and algorithms on a task of movie script classification. We reached the conclusion that more complex word representations are able to improve a model's performance by a small margin, when it comes to longer sequences of data, such as movie scripts.

In the future, we would like to explore the use of other types of contextual word embeddings as features. It would be interesting to investigate if pretrained embeddings of a model with more parameters than the distilled one would significantly improve the performance. On top of that, we would like to try more powerful models, other than Logistic Regression, for the classification.

Another thing we would like to try is to extract script structure information from the data, since they can be quite informative in certain genres. For instance, comedies mostly focus on dialogue to deliver jokes; as a result, the dialogue chunks would be far more than the chunks in an adventure film, which is action-packed and all action needs to be described thoroughly so that the action can be brought to life in the most successful manner. Because of time constraints, we did not develop a pipeline to extract structure information.

Finally, it would be interesting to investigate how the performance would be affected if training and testing was done on a bigger and more balanced dataset – the dataset we used had a class imbalance. Scraping movie scripts from the web and preprocessing all of them into a similar structure would be a good start to extending the already existing dataset.

References

John Arevalo, Thamar Solorio, Manuel Montes y Gómez, and Fabio A. González. 2017. Gated multimodal units for information fusion.

A. Blackstock and Matt Spitz. 2008. Classifying movie scripts by genre with a memm using nlp-based features.

Wei-Ta Chu and Hung-Jui Guo. 2017. Movie genre classification based on poster images with deep neural networks. In *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*, MUSA2 '17, page 39–45, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Ali Mert Ertugrul and Pinar KARAGOZ. 2018. Movie genre classification from plot summaries using bidirectional lstm.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Yusuke Nakano, Hiroaki Ohshima, and Yusuke Yamamoto. 2019. Film genre prediction based on film content and screenplay structure. In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications amp; Services*, iiWAS2019, page 151–155, New York, NY, USA. Association for Computing Machinery.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Howard Zhou, Tucker Hermans, Asmita Karandikar, and James Rehg. 2010. Movie genre classification via scene categorization. pages 747–750.