# Movie Genre Classification using Natural Language Processing

Authors: P. Soteriou, J. Evalle, S. Kok, S. Okuyan, A. Kocev

## Abstract

**Natural Language Processing (NLP) methods can be used to extract implicit information from text sources, providing insights about the features of those texts. In this project we used a variety of NLP methods to classify movies according to their genres, based upon the summaries of their plots. We collected and preprocessed the movie summaries and associated genres, and using bag-of-words and word embeddings approaches, we employed crude, namely naive Bayes, and refined machine learning methods, such as Support Vector Machines and Logistic Regression, for the task of text classification. We refined these methods by utilising filters and then compared the performance of the refined methods with those of more crude methods, such as the word counting and bag-of-words approaches. We found that the refined methods were better predictors of genre classification than the crude methods.**

## Introduction

The word ‘genre’ comes from French and originally means ‘a kind’. This meaning is kept to this day and it is used for the categorisation of literary works, such as books, music, TV series and movies. The movie genre determination takes into account four main features: story, plot, character and setting. Nonetheless, many of the genre definitions are not deterministic and they function more as guidelines because these classes overlap with each other (Chandler, 1997). For example, a movie containing multiple scenes of non-stop motion, dynamic scenes is categorised as action. However, that particular action movie may contain scenes that fit another category and therefore movies are generally described with multiple genres, including one that is the most exhaustive (Battu et al., 2018). The large number of genres and their corresponding guidelines make the genre determination process laborious. Our project seeks to develop an automated movie genre predictor using supervised Machine Learning methods based on movie summaries. A broader scope of this project is to use natural language processing (NLP) methods and text classification to accurately extract information and insights contained in texts and to categorise them (Bird et al., 2009).

To perform machine learning operations using written natural language data, we use the bag-of-words approach after preprocessing the data: a common tool of natural language processing. In this approach, text is represented in a way that disregards grammar, punctuation and word order, but focuses more on the multiplicity of the individual words and their general context (Youngjoong Ko, 2012). One effective technique of the bag of words approach we used is term frequency-inverse document frequency (TF-IDF) which is a measure that reflects the relevance of a word to a specific document (Rajaraman et al, 2011). The TF-IDF value of a word is proportional to the number of instances of that word in the document, but is also standardised by the number of documents that contain that word. As such, TF-IDF allows us to approximate the impact of a word on the classification of the document it is found within.

As a more elaborate approach, we also utilised word embedding, which converts words into a vector representation in a multidimensional (100D) space. This allows us to learn from target sets of featureless objects, such as words. Vector representation of words allows their numeric comparison with other words, as well as to operate on the words as arrays in methods that employ linear algebra.
We used three separate, standard classification Machine Learning methods to achieve our classification tasks, and then compared their performance.

The first method we used is a self-created method that makes use of the word counts per genre, called the Word Count Method (WCM). The second method involves the multinomial naive Bayes method (NBM) and uses an accumulated probability obtained from each word to predict the genre. The third method we used is a TF-IDF model paired with Complement naive Bayes classifier (Aizawa, 2003). Complement naive Bayes assigns weights in a different way than Multinomial naive Bayes does: training data is taken from all classes except the solution class and assigns the class that most poorly matches this complementary data (Rennie et al, 2003). The fourth method used is a Support Vector Machine (SVM), which is a method that builds decision boundaries between the separate classes. It does this based on the principle of support vectors, which alone can describe the separation of the classes in the model (Chang et al, 2011). The fifth method used is a Logistic Regression (LR) method, that modelled the probability of the separate classes of genre being assigned to the specific instances, defined by the descriptions of the movies (Vabalas A., 2019). Though normally suited for binary classification tasks, we nonetheless tested the model's ability to extend to our multiclass problem.

The performance metrics that were used are precision and recall. The genre classification problem is not a binary problem but a multiclass problem. Therefore, there is not just one confusion matrix, but as many as there are genres (Michie et al, 1994). While classifying an instance in the test set, there are two situations possible. If the classification is correct, the TP (true positive) cell in the confusion matrix of the correct label will increment by one. On the other hand, if the classification is incorrect, the FN (false negative) cell in the confusion matrix of the correct label will increment by one, while the FP (false positive) cell in the confusion matrix of the wrongly predicted genre will also increment by one.
This has two main impacts: there are no true negatives in any confusion matrix, and accuracy and recall, therefore, have the same value for each genre. Accuracy is (TP+TN)/total predictions, but since there are no TNs, it is calculated by dividing the true positives by the total number of predictions. Recall is TP/(TP+FN), but because either TP or FN increments for every instance TP+FN is equal to the number of total predictions, which also makes recall equal to the true positives by the total number of predictions. Therefore, accuracy was not used as a separate performance measure.

The aim of the project is to investigate whether using more refined word embedding model approaches, such as SVM and LR, outperform more basic bag-of-words methods in genre prediction, given movie summaries as input. We hypothesise that the more refined methods will have a better performance compared to the bag-of-words methods.

# Methods

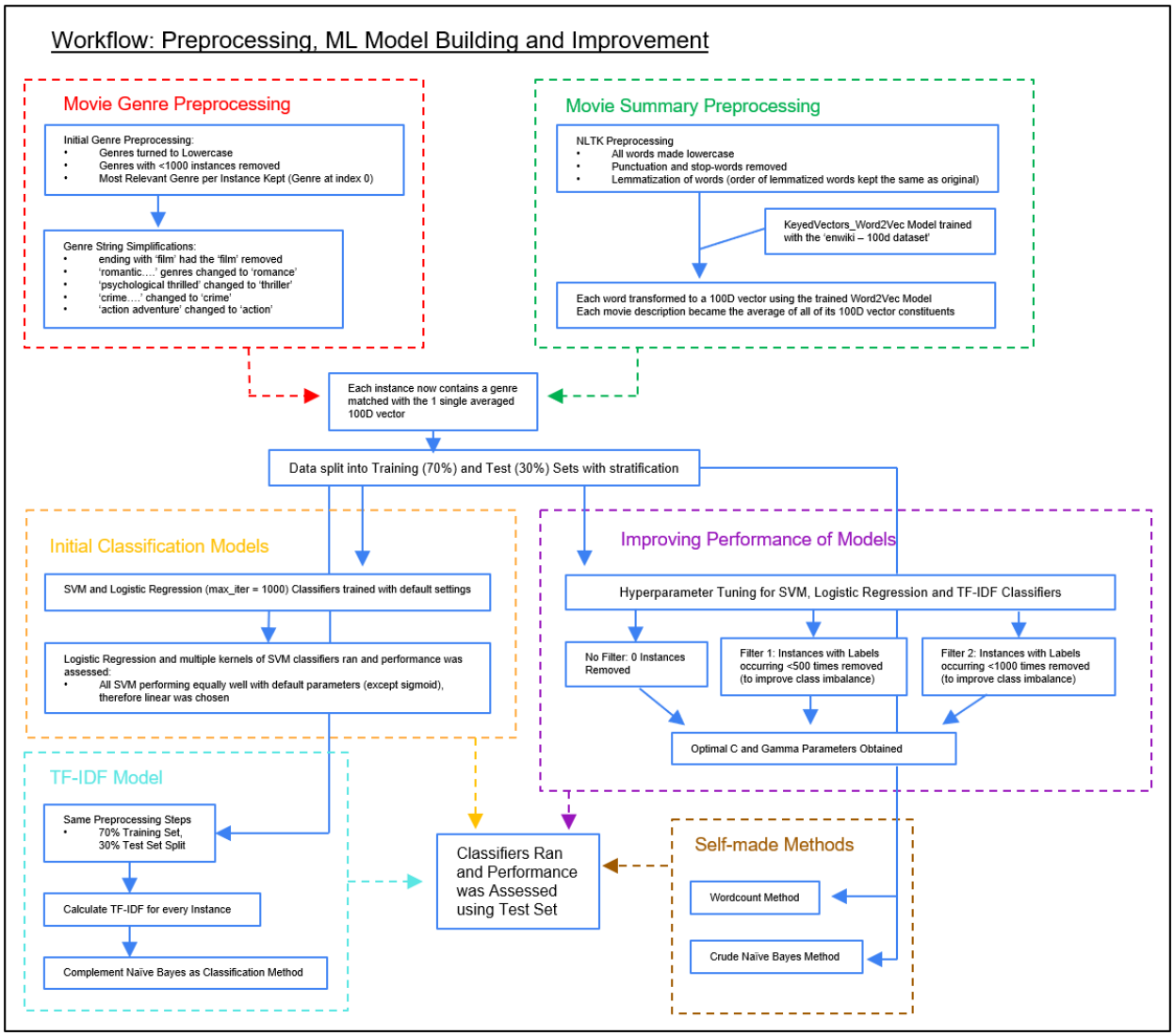


**Figure 1**: Project workflow showing the major steps involved in data preprocessing, model building and model improvement.

## Data Preprocessing

The datasets used, containing movie genres and summaries, were obtained from the published database of Carnegie Mellon University (Bamman et al. 2013). Data was obtained in .tsv file format, and merged by movie IDs. Irrelevant features (such as date, language, country, etc) were dropped. The features left in the dataset were movie ID, title, genrpe and description. Duplicate instances (of films) were removed. Any remaining punctuation that was present in the words was removed, as well as links to websites. All words within the genre and description features were turned to lowercase.

For genre-specific preprocessing, certain supervised modifications were required. Some genres were nearly identical to others, or were compounded, such as “romantic fantasy”. For any genres that contained “romantic”, they were changed to “romance”. For all such changes, see Figure 1 - Genre String

Simplifications. The total number of occurrences for every genre were counted. Genres occurring less than 1000 times were removed. If at this point, if an instance had no more genres, this instance was also removed. If a movie had several genres associated with it, all except the first one were removed, as the first genre in the list was the most representative. At the end, we were left with a total of 38744 instances and 26 distinct genres.

For description-specific preprocessing, using the WordNet Lemmatizer from NLTK (Bird et al., 2009), two columns were added to the dataset. The first column contained a regular lemmatised version of the description and the second column contained sets of the lemmatised descriptions. These sets were without duplicate words and without specific order. To convert the movie descriptions into vectors, the genism module and KeyedVectors.load_word2vec were used (Goldberg et al, 2014). This created an embedding between a key, one for each word, and an array of vectors. Word embedding methods are considered a refined method, as they are able to capture the context of a word in a text, the semantic and syntactic similarity, and its relation to other words. Pre-trained embeddings from enwiki_20180420 (window=5, iteration=10, negative=15) were used to obtain 100-dimensional vectors corresponding to single words (Mikolov et al, 2013). The parameter 'window' indicates that a distance of 5 words between the target word and the context to be predicted was taken into consideration when representing words as vectors. Iteration indicates that Wikipedia pages were iterated over 10 times to further refine the vector representation. Negative defines the number of negative samples (Yamada et al. 2016).

By iterating through the mapping (embeddings), words in the movie descriptions were converted to arrays made up of 100D-vector-entries corresponding to each word. Descriptions were then averaged to create an averaged 100D vector per instance, matched to a single genre.

To use the TF-IDF for text classification, the entire vocabulary should be known. Therefore, the TfidfVectorizer from sklearn was used on the entire dataset with descriptions. The classifier that made use of TF-IDF was Complement naive Bayes, which is especially useful for imbalanced datasets (Kibriya et al, 2004).

## Machine Learning Algorithms

Once the dataset consisted of single genres matched to their corresponding averaged 100D vector, it was split into 70% corresponding to the training set and 30% test set. There was stratification on all the genres of the dataset, to ensure equal representation of the labels in the training and test sets, using the scikit learn module and the train_test_split() function. Using this dataset, a selection of different machine learning methods was employed for the classification task.

### Wordcount Method

To create a baseline method we implemented our self-made method the WCM. For each instance in the training set, we kept track of which words were present in its summary and its genre. By assigning each word the genre the word appeared in the most, each word becomes associated with that genre. Then, during testing each word within the instance gets replaced by its associated genre. For example, the summary of [funny, exploded, violent] would be turned into [Comedy, Action, Action]. Every word that was not present in the training set was ignored. After this, the most common genre in the list will be chosen as the prediction for the instance. This process was repeated for each instance in the test set, with a random split in training and test set each iteration.

### Naive Bayes

The second method we implemented manually involved the use of the naive Bayes theorem, wherein each individual word got a certain predictive probability for each genre. During training, the algorithm went through the training set to obtain the three terms for the formula, shown in Formula 1.

$$p(genre|word) = \frac{p(word|genre) * p(genre)}{p(word)}$$

**Formula 1:** Formula calculating the posterior of the genre given a specific word.

First, the prior, representing the probability of seeing a specific genre, would be obtained from its fraction from all genres found. In the same vein, the evidence, the probability of observing a specific word, is obtained from its fraction from all the words found. The probability for a word given a genre is obtained by keeping track of the amount of times a certain genre was found with the word in question. For all of the genres per word, there was a pseudo-count of 1 incremented to ensure that the final probability can be calculated. Then, each pseudo-counted genre for the word was divided by the total amount of genre counts for that word to get the likelihood for that word given a genre. During testing, the algorithm goes through each instance in the test set and calculates the probability of each genre. This is done by going through each word of the instance for each genre found in the training set. For each word in the summary, we calculated the posterior for the current genre-word combination and used the sum-log function to obtain a final probability from all of the posteriors. E.g. The final probability summary [funny, exploded, violent] for the genre action would be calculated as ∑log([0.004, 0.012, 0.016]). Sum-log is used as a substitution for the product to calculate the final probability due to the amount of words a summary can contain, taking the combined product of all of them could result in arithmetic underflow. The genre with the highest probability was then chosen as the predicted genre for the instance.

Only the WCM and the NBM will make use of multiple iterations due to their short run time, with each iteration using a random split between training and test set each iteration.

### SVM and Logistic Regression Methods

Classifiers with multiple SVM kernels (linear, polynomial with 5, 7 and 10 degrees, radial basis function and sigmoid) and logistic regression with the maximum iterations parameter (max_iter) set to 1000 were tested with their default parameters and were used to define a baseline and evaluate default model performance.

## Performance Improvement and Testing

Aiming to further improve our classifier performance, we chose the linear kernel for the SVM, as it is the simplest and performed equally well as the more complex models. For the SVM and LR classifiers, we applied gridsearchCV() with each of our models. Grid search works by optimising the parameters of the estimator via cross-validating over a parameter grid. For the SVM, the parameters were C (0, 1, 10, 100) and gamma (1, 0.1, 0.01, 0.001, 0.0001). GridsearchCV() was also applied for the LR model using parameters C (-5, 8 and 15) and cv (5) (cross validation, the number of cv folds for each combination of parameters). The classifiers were then retrained using the optimal parameters and the new performance was obtained.

The C parameter is a penalty parameter, which represents misclassification or the error term, and informs the SVM optimisation how much error is acceptable. This allows the controlling of the trade-off between the decision boundary and the misclassification term. A high C will classify all the data correctly, with the

risk of overfitting. The gamma parameter influences the calculation of a plausible hyperplane/line of separation. A high gamma indicates that nearby points have high influence and low gamma indicates that far away points are also considered to get the decision boundary (Vabalas et al, 2019).

In an attempt to further improve our model performance and address class imbalance, we utilised two different filters. For these filters we removed instances with genres occurring <500 (filter 1) and <1000 (filter 2) times in the training set, and re-ran the hyperparameter optimisation for each filter, as described above. Hyperparameter tuning was only done for the SVM and LR classifiers, since naive Bayes used for the TF-IDF, WCM and NBM classifiers does not feature hyperparameters. In addition, a control method was implemented to define our baseline. This method assigned genres based on the majority class (drama).

Once all models were trained on the training set, their individual performances were tested using the test set (No filter: 11623 instances, Filter 1: 10753 instances, Filter 2: 8592 instances).

## Results

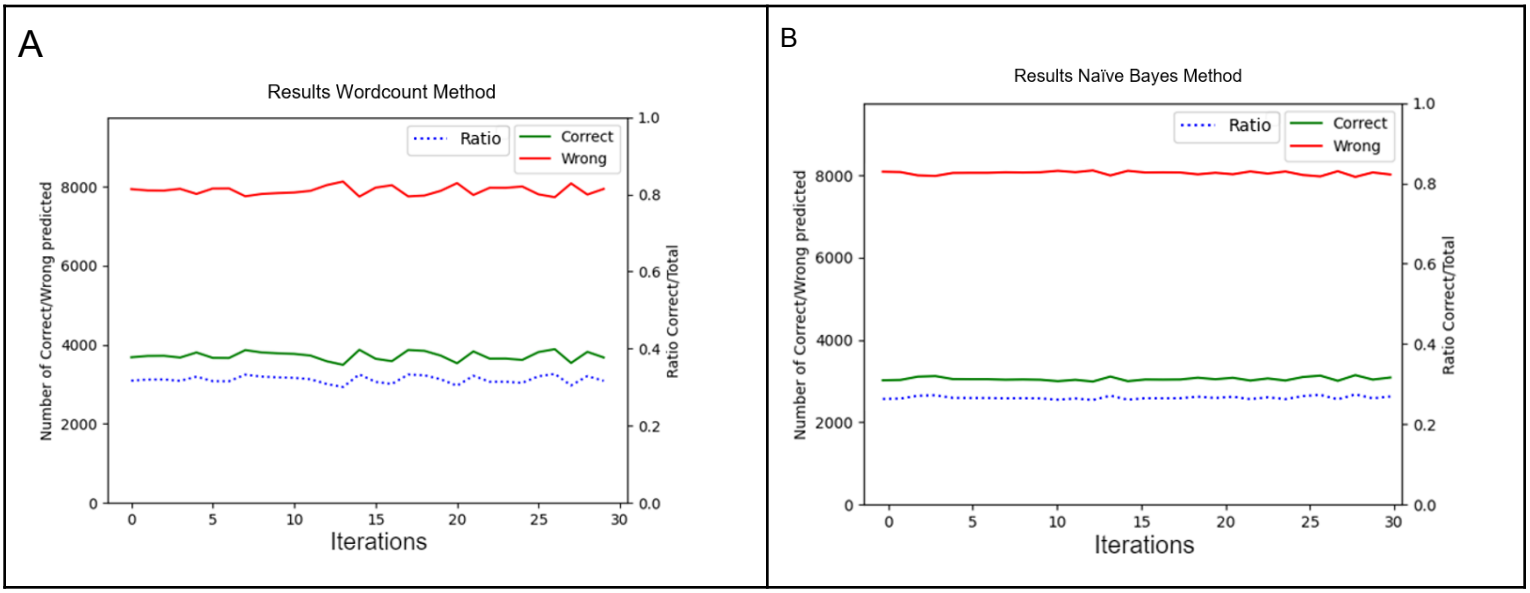


**Figure 2: Results Wordcount Method and Naive Bayes Method over 30 iterations**
Plots showing the amount of correctly (green) and wrongly (red) predicted instances in the test set of each iteration. The dotted blue line represents the fraction of the instances that were predicted correctly. Plot A contains the results of the Wordcount Method, showing that on average it predicts around 3800 instances correct, around 8000 instances wrong and has a correctness ratio of around 35%. Plot B contains the results of the naive Bayes Method, showing that on average it predicts around 3700 instances correct and 8100 instances wrong and has a correctness ratio of around 30%.

The performance results of the WCM and the NBM can be seen in Figure 2. In both, the predictive quality of the methods seem to be stable over the 30 iterations, with the WCM being less stable but generally predicting a larger fraction of the test set correctly when compared to the NBM, ~ 35% and 30% respectively.

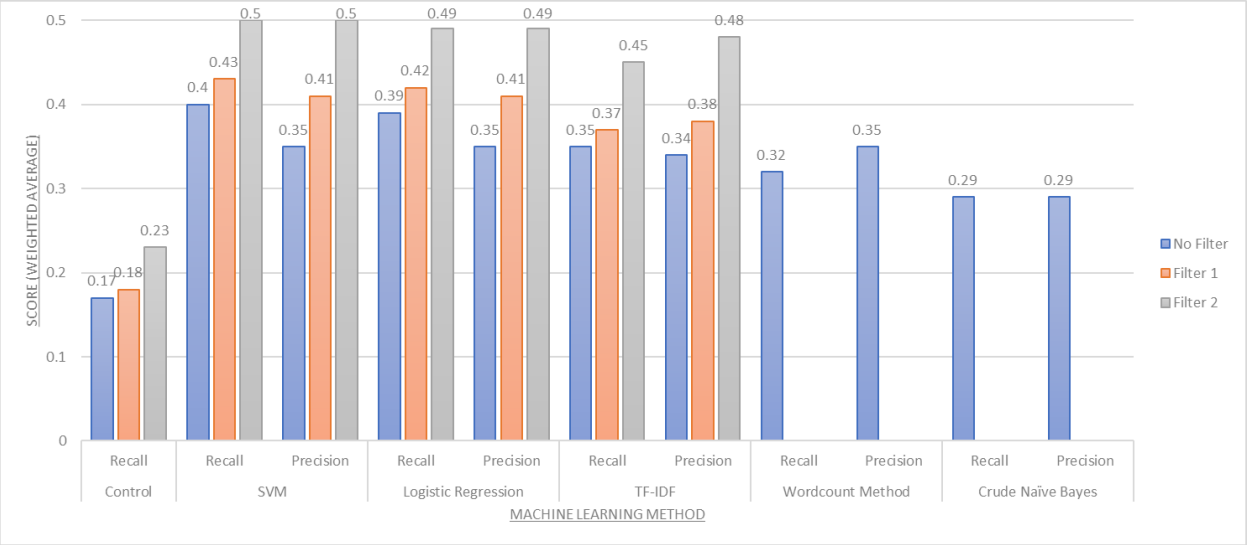


**Figure 3:** Model performances. The different machine learning models have their averaged recall and precisions compared against each other. The recall and precision scores are also compared between the different filters used within the same methods.

Figure 3 shows the averaged precision and averaged recall of the different methods when with the different types of filtering. As expected, filtering improved model performance and all methods outperformed random classification, which is the assignment of the majority class, i.e. drama (count = 1934), to all instances (No filter: 11623, Filter 1: 10753, Filter 2: 8592). As per our expectations, the more refined methods (SVM, LR and TF-IDF) outperformed the basic, baseline WCM and NBM.

No filtering yields the lowest overall performance for all models, shown by recall. The best overall model performance is produced by linear SVM with score of 0.4, followed by LR with score of 0.39. TF-IDF, WCM and NBM had the poorest performance with a score of 0.29. Precision is more stable, with values ranging from 0.35 to 0.29, and the model performance follows a similar trend to recall. Interestingly, precision was higher than recall for the WCM and equal to the recall for NBM.

Filter 1 modestly improves model performance, increasing recall for SVM and LR by 0.03 and 0.02 for TF-IDF. Correspondingly, precision increases and provides a larger effect than recall, up to 0.06 in SVM and LR. Filter 2 offers a more substantial improvement in model performance, increasing TF-IDF recall by 0.08 to 0.45 and SVM and LR by 0.07 to 0.5 and 0.49, respectively. Precision improves by 0.1 in TF-IDF and by 0.9 in SVM and 0.8 in LR.

## Discussion

All methods had higher recall than the random model which suggests that they managed to discover the correlation between plot summaries and genres. Word embedding methods with SVM and LR classifiers perform best when looking at precision and recall. Both methods performed equally well, TF-IDF performed slightly worse, the WCM had similar precision but worse recall and NBM performed the worst. This agrees with our initial expectation that more refined word embedding models outperform more basic bag-of-word methods for genre classification, given movie summaries. Also, it is important to state that filtering approaches increased relative performances up to 42%. However, none of the methods outperforms 50% in recall or precision. This could be explained by the composition of the used dataset and limitations of text representation techniques. First, we experienced class imbalance, particularly with the action and drama classes being overrepresented. It is likely that LR is more sensitive to class imbalance despite hyperparameter optimisation which could explain different performances, even though this difference is small.

One possible explanation as to the drama and action genres being overrepresented in the data because of their broad definition. Since these genres are very general and can encompass a wide range of movie types, they appear quite frequently, even when the movies they describe are not solely of that type (Chandler D., 1997). One could suggest using an oversampling technique to overcome the class imbalance but this would be much more complex than usual because each instance represents multiple genres and balancing that out would require excessive tinkering. Secondly, one potential problem of the bag-of-words method is that it does not take the word order into consideration. This is a challenge for NLP tasks since it can lead to the loss of semantics. For instance, the sentence ‘car is red’ has the same sentence embedding as ‘red is car’. Conversely, word embeddings would result in slightly different vector values reflecting the effect and nuance given by the word order, which is why they gave better results. In the word embedding approach, words which were not found in the enwiki_20180420 dataset were excluded from the word embedding. This is because words not found in the word2vec mapping have no vectors assigned to them and therefore, they cannot be represented in vector format. An additional limitation of the word2vec mapping is that of static word embedding, where each word is assigned to a single 100D vector. This could be problematic when embedding words with multiple meanings since the vector will represent the most frequent meaning of the word and not incorporate the whole range of definitions it could represent.

In agreement with our hypothesis and results, more refined methods outperform cruder methods. This was also exemplified by a Kaggle competition on NLP, which utilised recurrent neural networks and long short-term memory networks, where the top scoring model achieved a model performance/accuracy of 0.65 (Lekov, 2019). Because of this, we expect that other more refined methods, like neural networks or BERT, could further improve the genre classification. Another improvement to our methods would be to utilise dynamic word embedding. That will allow us to represent words and contexts by latent trajectories in an embedding space, thus allowing the word and context vectors to drift. Dynamic word embeddings were shown to have better predictive abilities than static word embeddings (Balmer & Mandt, 2017).

### Future research

In the future, we aim to expand our approach into a multilabel classification task. In this research only the principal genre of movies was predicted, but it is often the case that movies are defined by multiple genres, because genres have overlapping characteristics and are not mutually exclusive. As one would expect, plot summaries can also follow the same trend, by having words that may indicate two different genres. Therefore, multilabel classification may allow one to design models that predict with a higher accuracy; albeit they will be computationally more expensive. Another challenge for this project would be that not every movie can be described by the same number of genres: some would fall into five genres, while others could be described by three. It would be interesting to find the optimal number of genre labels to predict for this multilabel classification. Identifying the words that are strong predictors of genres could also be insightful. Also, the oversampling technique would be available to use which was not the case for our models. Aside from the saved labour and time, movie genre classification could be seen as a trivial task but using methods applied in this project for text classification could be appropriate in different contexts such as analysis of electronic healthcare records and diagnosis/prognosis.

## Conclusion

When predicting movie genres from summaries, the more refined methods, like SVM and LR, performed better than the less refined bag-of-word approaches, namely TF-IDF, WCM and NBM. This is most likely attributed to the use of word embeddings, which allows vector representation of words to encapsulate context and semantics, and hence more accurately portray information.

# References


Aizawa, A (2003). *"An information-theoretic perspective of tf–idf measures"*. Information Processing and Management. 39 (1): 45–65. doi:10.1016/S0306-4573(02)00021-3

Bamler R., Mandt S. (2017) *"Dynamic Word Embeddings"*. International Conference on Machine Learning. https://doi.org/10.48550/arXiv.1702.08359

Bamman D., O'Connor B., Smith N. (2013). *"Learning Latent Personas of Film Characters"* ACL. https://aclanthology.org/P13-1035.pdf

Battu V., Batchu V., Reddy R., Reddy M., Mamidi R. (2018). "*Predicting the Genre and Rating of a Movie Based on its Synopsis".* International Institute of Information Technology Hyderabad. PACLIC 32. https://aclanthology.org/Y18-1007.pdf

Bird S., Klein E., Loper E., (2009). "*Natural Language Processing with Python"*. O'Reilly Media Inc.

Chandler, D (1997). *"An Introduction to Genre Theory"*. http://www.aber.ac.uk/media/Documents/intgenre/chandler_genre_theory.pdf

Chang C., Lin CJ. (2011). *"LIBSVM: A library for support vector machines".* ACM Transactions on Intelligent Systems and Technology (TIST). https://dl.acm.org/doi/pdf/10.1145/1961189.1961199

Goldberg Y., Levy, O. (2014). *"word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method"*. Cornell University. https://doi.org/10.48550/arXiv.1402.3722

Kibriya A.M., Frank E., Pfahringer B., Holmes G. (2004). *"Multinomial Naive Bayes for Text Categorization"*. Advances in Artificial Intelligence. https://doi.org/10.1007/978-3-540-30549-1_43

Lekov A. (2019)."fastai Text Classification". https://www.kaggle.com/code/itslek/fastai-text-classification-movie-genre-sf-v1-8/notebook?scriptVersionId=13406083

Michie, D., Spiegelhalter D., Taylor C. C. (1994). *"Machine Learning, Neural and Statistical Classification"*. Ellis Horwood Series in Artificial Intelligence.

Mikolov T., Chen K., Corrado G., Dean J. (2013). "*Efficient Estimation of Word Representations in Vector Space"*. Cornell University. https://doi.org/10.48550/arXiv.1301.3781

Rajaraman A., Ullman J. (2011). *"Data Mining"*. Mining of Massive Datasets. pp. 1–17. https://doi:10.1017/CBO9781139058452.002

Rennie J., Shih L., Teevan J., Karger D. (2003). *"Tackling the Poor Assumptions of Naive Bayes Text Classifiers".* AAAI Press. http://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf

Vabalas A., Gowen E., Poliakoff E., Casson A. (2019). " *Machine learning algorithm validation with a limited sample size".* PLOS ONE. https://doi.org/10.1371/journal.pone.0224365

Yamada I., Shindo H., Takeda H., Takefuji Y. (2016). "*Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation".* Association for Computational Linguistics. https://doi.org/10.48550/arXiv.1601.01343

Youngjoong Ko (2012). *"A study of term weighting schemes using class information for text classification"*. *SIGIR'12*. ACM. https://dl.acm.org/doi/pdf/10.1145/2348283.2348453

# Appendix

Wordcount method results

| | Precision | Recall | Support |
|---|---|---|---|
| Drama | 0.29 | 0.91 | 811442 |
| Thriller | 0.37 | 0.46 | 71748 |
| Romance | 0.48 | 0.00 | 42237 |
| Short | 0.65 | 0.03 | 25264 |
| Crime | 0.68 | 0.00 | 21594 |
| Comedy | 0.00 | 0.00 | 19702 |
| Action | 0.00 | 0.00 | 15203 |
| Horror | 0.00 | 0.00 | 13872 |
| Science Fiction | 1.00 | 0.00 | 7807 |
| Family | 0.64 | 0.00 | 5889 |
| Musical | 0.00 | 0.00 | 5805 |
| Documentary | 0.78 | 0.00 | 5685 |
| Adventure | 0.00 | 0.00 | 5624 |
| World Cinema | 0.00 | 0.00 | 5031 |
| Japanese Movies | 0.00 | 0.00 | 3623 |
| Indie | 0.00 | 0.00 | 3579 |
| Period Piece | 1.00 | 0.00 | 3555 |
| Comedy-Drama | 0.00 | 0.00 | 2556 |
| Fantasy | 0.00 | 0.00 | 2291 |
| Black-and-White | 1.00 | 0.00 | 1692 |
| Mystery | 0.00 | 0.00 | 1374 |
| Children | 0.00 | 0.00 | 976 |
| Animaton | 0.00 | 0.00 | 203 |
| War | 0.00 | 0.00 | 44 |
| Film Adaptation | 0.00 | 0.00 | 19 |

Naive Bayes Method results

| | Precision | Recall | Support |
|---|---|---|---|
| Drama | 0.26 | 0.97 | 58137 |
| Thriller | 0.30 | 0.89 | 45510 |
| Romance | 0.93 | 0.07 | 41204 |
| Short | 0.88 | 0.01 | 26068 |
| Crime | 0.00 | 0.00 | 25615 |
| Comedy | 0.00 | 0.00 | 25307 |
| Action | 0.50 | 0.00 | 23076 |
| Horror | 0.00 | 0.00 | 14688 |
| Science Fiction | 0.00 | 0.00 | 11364 |
| Family | 0.00 | 0.00 | 9675 |
| Musical | 0.00 | 0.00 | 9371 |
| Documentary | 0.00 | 0.00 | 7226 |
| Adventure | 0.00 | 0.00 | 7054 |
| World Cinema | 0.00 | 0.00 | 6396 |
| Japanese Movies | 0.00 | 0.00 | 6375 |
| Indie | 0.00 | 0.00 | 6166 |
| Period Piece | 0.00 | 0.00 | 5926 |
| Comedy-Drama | 0.00 | 0.00 | 2889 |
| Fantasy | 0.00 | 0.00 | 2664 |
| Black-and-White | 0.00 | 0.00 | 2453 |
| Mystery | 0.00 | 0.00 | 2426 |
| Children | 0.00 | 0.00 | 1943 |
| Animaton | 0.00 | 0.00 | 1674 |
| War | 0.00 | 0.00 | 772 |
| Film Adaptation | 0.00 | 0.00 | 441 |

# Hyperparameter tuning

**Hyperparameter tuning/optimisation results**

No Filter

| Model | Accuracy |
|---|---|
| SVM no filtering | |
| model = LinearSVC() | 0.39 |
| model = SVC(kernel = 'linear') | 0.4 |
| model = SVC(kernel = 'poly', degree = 5) | 0.4 |
| model = SVC(kernel = 'poly', degree = 7) | 0.4 |
| model = SVC(kernel = 'poly', degree = 10) | 0.4 |
| model = SVC(kernel = 'rbf') | 0.4 |
| model = SVC(kernel = 'sigmoid') | 0.37 |
| | |
| optimised parameters (obtained from hyperparameter tuning): | |
| SVC(C=10, gamma=1, kernel='linear') | 0.4 |
| | |
| Log regression no filtering | |
| model = LogisticRegression() | 0.39 |
| | |
| optimised parameters (obtained from hyperparameter tuning): | 0.39 |
| Tuned Logistic Regression Parameters: {'C': 3.727593720314938} | |
| Best score is 0.38923156471036646 | |

Filter 1 - Removing labels w/ <500 occurrences in the training set

| All trained with stratified = gold_genre ; | Accuracy | |
|---|---|---|
| SVM w/ filtering | | filtering = instances w/ labels occuring <500 times in the training dataset are removed |
| optimised parameters (obtained from hyperparameter tuning): | | |
| SVC(C=10, gamma=1, kernel='linear') | 0.43 | |
| Log regression w/ filtering | | |
| model = LogisticRegression(max_iter=1000) | 0.42 | |
| optimised parameters (obtained from hyperparameter tuning): | | |
| Tuned Logistic Regression Parameters: {'C': 31.622776601683793} | 0.42 | |
| Best score is 0.42254908752715065 | | |

Filter 2 - Removing labels w/ <1000 occurrences in the training set

| | Accuracy | |
|---|---|---|
| SVM w/ filtering_2 | | filtering = instances w/ labels occuring <1000 times in the training dataset are removed |
| optimised parameters (obtained from hyperparameter tuning): | | |
| SVC(C=1, gamma=1, kernel='linear') | 0.5 | |
| Log regression w/ filtering_2 | | |
| model = LogisticRegression(max_iter=1000, C=2275.845926074791) | 0.49 | |
| optimised parameters (obtained from hyperparameter tuning): | | |
| Tuned Logistic Regression Parameters: {'C': 2275.845926074791} | | |
| Best score is 0.4982715352544499 | | |

# SVM - 3 Filter results- maybe confusion matrix or something else to show these more intuitively?

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

----------SVM LINEAR Classifier Report----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| "children | 0.00 | 0.00 | 0.00 | 68 |
| action | 0.35 | 0.41 | 0.38 | 702 |
| adventure | 0.27 | 0.18 | 0.21 | 259 |
| animation | 0.00 | 0.00 | 0.00 | 64 |
| black-and-white | 0.00 | 0.00 | 0.00 | 81 |
| comedy | 0.28 | 0.22 | 0.25 | 853 |
| comedy-drama | 0.00 | 0.00 | 0.00 | 97 |
| crime | 0.38 | 0.31 | 0.34 | 856 |
| documentary | 0.45 | 0.55 | 0.49 | 203 |
| drama | 0.36 | 0.56 | 0.44 | 1934 |
| family | 0.26 | 0.11 | 0.16 | 319 |
| fantasy | 0.08 | 0.01 | 0.02 | 91 |
| film adaptation | 0.00 | 0.00 | 0.00 | 14 |
| horror | 0.47 | 0.39 | 0.43 | 493 |
| indie | 0.00 | 0.00 | 0.00 | 198 |
| japanese movies | 0.55 | 0.55 | 0.55 | 214 |
| musical | 0.23 | 0.06 | 0.09 | 310 |
| mystery | 0.00 | 0.00 | 0.00 | 80 |
| period piece | 0.00 | 0.00 | 0.00 | 122 |
| romance | 0.48 | 0.54 | 0.46 | 1372 |
| science fiction | 0.43 | 0.43 | 0.43 | 376 |
| short | 0.56 | 0.57 | 0.57 | 869 |
| silent | 0.25 | 0.11 | 0.15 | 233 |
| thriller | 0.42 | 0.54 | 0.48 | 1512 |
| war | 0.00 | 0.00 | 0.00 | 56 |
| world cinema | 0.00 | 0.00 | 0.00 | 247 |
| accuracy | | | 0.40 | 11623 |
| macro avg | 0.22 | 0.21 | 0.21 | 11623 |
| weighted avg | 0.35 | 0.40 | 0.36 | 11623 |

```
model = SVC(kernel='linear', C=10, gamma=1)
```

----------SVM LINEAR Classifier Report----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.36 | 0.39 | 0.38 | 702 |
| adventure | 0.30 | 0.19 | 0.23 | 259 |
| comedy | 0.32 | 0.21 | 0.26 | 853 |
| crime | 0.39 | 0.34 | 0.36 | 856 |
| documentary | 0.50 | 0.56 | 0.53 | 203 |
| drama | 0.40 | 0.56 | 0.46 | 1934 |
| family | 0.38 | 0.15 | 0.22 | 319 |
| horror | 0.51 | 0.42 | 0.46 | 493 |
| japanese movies | 0.49 | 0.49 | 0.49 | 214 |
| musical | 0.27 | 0.09 | 0.14 | 310 |
| romance | 0.43 | 0.55 | 0.48 | 1373 |
| science fiction | 0.47 | 0.43 | 0.45 | 376 |
| short | 0.62 | 0.58 | 0.60 | 869 |
| silent | 0.40 | 0.14 | 0.21 | 233 |
| thriller | 0.44 | 0.51 | 0.47 | 1512 |
| world cinema | 0.00 | 0.00 | 0.00 | 247 |
| accuracy | | | 0.43 | 10753 |
| macro avg | 0.39 | 0.35 | 0.36 | 10753 |
| weighted avg | 0.41 | 0.43 | 0.41 | 10753 |

```
model = SVC(kernel='linear', C=10, gamma=1)
```

```
{'C': 1, 'gamma': 1, 'kernel': 'linear'}
SVC(C=1, gamma=1, kernel='linear')
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.47 | 0.43 | 0.45 | 702 |
| comedy | 0.39 | 0.20 | 0.26 | 853 |
| crime | 0.41 | 0.31 | 0.35 | 856 |
| drama | 0.49 | 0.60 | 0.54 | 1934 |
| horror | 0.57 | 0.34 | 0.43 | 493 |
| romance | 0.51 | 0.55 | 0.53 | 1373 |
| short | 0.72 | 0.68 | 0.70 | 869 |
| thriller | 0.46 | 0.58 | 0.51 | 1512 |
| accuracy | | | 0.50 | 8592 |
| macro avg | 0.50 | 0.46 | 0.47 | 8592 |
| weighted avg | 0.50 | 0.50 | 0.49 | 8592 |

Optimal model parameters were obtained using GridSearchCV(SVC(), …). Grid parameters tested: 'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], kernel = ['linear']. I have yet to figure out what parameters C and gamma are

No filter, filter 1 and filter 2 (SVM) random: 1934/11623, 1934/10753, 1934/8592

## Log Regr - 3 Filter results - maybe confusion matrix or something else to show these more intuitively?

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

----------LOG REGRESSION Classifier Report----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| "children | 0.33 | 0.04 | 0.08 | 68 |
| action | 0.36 | 0.37 | 0.36 | 702 |
| adventure | 0.29 | 0.15 | 0.20 | 259 |
| animation | 0.00 | 0.00 | 0.00 | 64 |
| black-and-white | 0.00 | 0.00 | 0.00 | 81 |
| comedy | 0.30 | 0.20 | 0.24 | 853 |
| comedy-drama | 0.00 | 0.00 | 0.00 | 97 |
| crime | 0.39 | 0.30 | 0.34 | 856 |
| documentary | 0.47 | 0.46 | 0.47 | 203 |
| drama | 0.37 | 0.56 | 0.44 | 1934 |
| family | 0.26 | 0.12 | 0.16 | 319 |
| fantasy | 0.13 | 0.02 | 0.04 | 91 |
| film adaptation | 0.00 | 0.00 | 0.00 | 14 |
| horror | 0.46 | 0.40 | 0.43 | 493 |
| indie | 0.00 | 0.00 | 0.00 | 198 |
| japanese movies | 0.55 | 0.41 | 0.47 | 214 |
| musical | 0.22 | 0.07 | 0.11 | 310 |
| mystery | 0.00 | 0.00 | 0.00 | 80 |
| period piece | 0.18 | 0.02 | 0.03 | 122 |
| romance | 0.40 | 0.54 | 0.46 | 1372 |
| science fiction | 0.44 | 0.40 | 0.42 | 376 |
| short | 0.52 | 0.60 | 0.56 | 869 |
| silent | 0.26 | 0.14 | 0.18 | 233 |
| thriller | 0.40 | 0.56 | 0.47 | 1512 |
| war | 0.06 | 0.02 | 0.03 | 56 |
| world cinema | 0.09 | 0.01 | 0.02 | 247 |
| accuracy | | | 0.39 | 11623 |
| macro avg | 0.25 | 0.21 | 0.21 | 11623 |
| weighted avg | 0.35 | 0.39 | 0.36 | 11623 |

```
model = LogisticRegression(max_iter=1000, C=3.7276)
```

----------LOG REGRESSION Classifier Report----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.38 | 0.36 | 0.37 | 702 |
| adventure | 0.31 | 0.20 | 0.24 | 259 |
| comedy | 0.33 | 0.20 | 0.25 | 853 |
| crime | 0.39 | 0.30 | 0.33 | 856 |
| documentary | 0.54 | 0.50 | 0.52 | 203 |
| drama | 0.40 | 0.55 | 0.47 | 1934 |
| family | 0.35 | 0.16 | 0.22 | 319 |
| horror | 0.48 | 0.43 | 0.45 | 493 |
| japanese movies | 0.52 | 0.39 | 0.45 | 214 |
| musical | 0.26 | 0.12 | 0.16 | 310 |
| romance | 0.42 | 0.55 | 0.48 | 1373 |
| science fiction | 0.45 | 0.39 | 0.42 | 376 |
| short | 0.57 | 0.59 | 0.58 | 869 |
| silent | 0.38 | 0.20 | 0.26 | 233 |
| thriller | 0.42 | 0.53 | 0.47 | 1512 |
| world cinema | 0.06 | 0.01 | 0.01 | 247 |
| accuracy | | | 0.42 | 10753 |
| macro avg | 0.39 | 0.34 | 0.35 | 10753 |
| weighted avg | 0.41 | 0.42 | 0.40 | 10753 |

```
model = LogisticRegression(max_iter=1000, C=31.6228)
```

----------LOG REGRESSION Classifier Report----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.46 | 0.40 | 0.43 | 702 |
| comedy | 0.37 | 0.22 | 0.27 | 853 |
| crime | 0.42 | 0.31 | 0.36 | 856 |
| drama | 0.48 | 0.57 | 0.52 | 1934 |
| horror | 0.53 | 0.39 | 0.45 | 493 |
| romance | 0.51 | 0.55 | 0.53 | 1373 |
| short | 0.69 | 0.69 | 0.69 | 869 |
| thriller | 0.46 | 0.56 | 0.51 | 1512 |
| accuracy | | | 0.49 | 8592 |
| macro avg | 0.49 | 0.46 | 0.47 | 8592 |
| weighted avg | 0.49 | 0.49 | 0.48 | 8592 |

```
model = LogisticRegression(max_iter=1000, C=2275.845926074791)
```

Optimal model parameters were obtained using GridSearchCV(logreg, param_grid, cv = 5), c_space = np.logspace(-5, 8, 15) and param_grid = {'C': c_space}. I have yet to figure out what parameters C and cv are.

## TF-IDF results

### Unfiltered

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.36 | 0.20 | 0.26 | 702 |
| adventure | 0.28 | 0.09 | 0.14 | 259 |
| animation | 0.50 | 0.08 | 0.14 | 64 |
| black-and-white | 0.00 | 0.00 | 0.00 | 81 |
| children's/family | 0.12 | 0.01 | 0.03 | 68 |
| comedy | 0.35 | 0.07 | 0.12 | 853 |
| comedy-drama | 0.00 | 0.00 | 0.00 | 97 |
| crime | 0.36 | 0.12 | 0.18 | 856 |
| documentary | 0.43 | 0.11 | 0.18 | 203 |
| drama | 0.31 | 0.65 | 0.42 | 1934 |
| family | 0.25 | 0.05 | 0.08 | 319 |
| fantasy | 0.09 | 0.01 | 0.02 | 91 |
| film adaptation | 0.00 | 0.00 | 0.00 | 14 |
| horror | 0.49 | 0.19 | 0.28 | 493 |
| indie | 0.00 | 0.00 | 0.00 | 198 |
| japanese movies | 0.58 | 0.33 | 0.42 | 214 |
| musical | 0.12 | 0.01 | 0.02 | 310 |
| mystery | 0.33 | 0.01 | 0.02 | 80 |
| period piece | 0.00 | 0.00 | 0.00 | 122 |
| romance | 0.35 | 0.46 | 0.40 | 1373 |
| science fiction | 0.47 | 0.34 | 0.40 | 376 |
| short | 0.57 | 0.62 | 0.59 | 869 |
| silent | 0.39 | 0.05 | 0.09 | 233 |
| thriller | 0.32 | 0.65 | 0.43 | 1512 |
| war | 0.00 | 0.00 | 0.00 | 56 |
| world cinema | 0.03 | 0.00 | 0.01 | 247 |
| accuracy | | | 0.35 | 11624 |
| macro avg | 0.26 | 0.16 | 0.16 | 11624 |
| weighted avg | 0.34 | 0.35 | 0.30 | 11624 |

### Filter 1

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.38 | 0.20 | 0.26 | 702 |
| adventure | 0.34 | 0.09 | 0.15 | 259 |
| children's/family | 0.00 | 0.00 | 0.00 | 68 |
| comedy | 0.41 | 0.08 | 0.14 | 853 |
| crime | 0.36 | 0.13 | 0.19 | 857 |
| documentary | 0.57 | 0.13 | 0.22 | 203 |
| drama | 0.34 | 0.66 | 0.45 | 1934 |
| family | 0.38 | 0.05 | 0.09 | 319 |
| horror | 0.56 | 0.21 | 0.31 | 493 |
| japanese movies | 0.53 | 0.29 | 0.37 | 214 |
| musical | 0.19 | 0.02 | 0.03 | 310 |
| romance | 0.35 | 0.44 | 0.39 | 1373 |
| science fiction | 0.49 | 0.35 | 0.41 | 376 |
| short | 0.59 | 0.60 | 0.60 | 869 |
| silent | 0.37 | 0.03 | 0.06 | 233 |
| thriller | 0.34 | 0.63 | 0.44 | 1512 |
| world cinema | 0.07 | 0.01 | 0.02 | 247 |
| accuracy | | | 0.37 | 10822 |
| macro avg | 0.37 | 0.23 | 0.24 | 10822 |
| weighted avg | 0.38 | 0.37 | 0.33 | 10822 |

### Filter 2

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| action | 0.54 | 0.21 | 0.31 | 702 |
| children's/family | 0.44 | 0.06 | 0.10 | 68 |
| comedy | 0.53 | 0.07 | 0.12 | 853 |
| crime | 0.40 | 0.11 | 0.18 | 856 |
| drama | 0.41 | 0.68 | 0.51 | 1934 |
| horror | 0.64 | 0.19 | 0.30 | 493 |
| romance | 0.44 | 0.44 | 0.44 | 1373 |
| short | 0.74 | 0.63 | 0.68 | 869 |
| thriller | 0.40 | 0.67 | 0.50 | 1512 |
| accuracy | | | 0.45 | 8660 |
| macro avg | 0.50 | 0.34 | 0.35 | 8660 |
| weighted avg | 0.48 | 0.45 | 0.41 | 8660 |