Movie Genres Prediction

DAVID MIHOLA (12211951) and JOHANNES PESENHOFER (11771884)

ABSTRACT

This project focuses on creating a movie script classification system using web crawling and machine learning techniques. The first step involves collecting movie scripts from different sources on the internet, specifically Imsdb[1] and Dailyscript[2] websites. These scripts are then cleaned, pre-processed and splitted into training and test data sets.

Next, multiple machine learning models were implemented for classification. One of the models is based on simple statistics, counting the occurrence of words to predict the genre of a script. The other models are based on a combination of machine learning techniques, namely combining transformer architectures for text embeddings generation and fully connected neural networks for classification. These models were trained using the prepared data sets.

We evaluate and compare the performance of the different models with metrics including, precision, recall, intersection over union and F1-score.

Additionally, a graphical user interface (GUI) was developed to provide an easy way for the users to interact with the movie script classification system. Users can input movie scripts and receive predicted genres inferred by the trained models. Moreover, the GUI enables the users to scrape the data of a given website, pre-process them, train models on the scraped data sets and evaluate them.

Additional Key Words and Phrases: movie genres prediction, web-scraping, sentence transformers, embeddings, SBERT, BERT, GloVe, uni-gram

ACM Reference Format:

David Mihola (12211951) and Johannes Pesenhofer (11771884). 2023. Movie Genres Prediction. , 24 pages.

[1] https://imsdb.com
[2] https://www.dailyscripts.com

Manuscript submitted to ACM

Contents

1 INTRODUCTION

In recent years movie and tv-show streaming platforms started to become more and more popular. In order to give proper recommendations for users it became crucial for these platforms to provide meaningful and precise tags for their titles. One form of tags for movies and tv-shows are genres. This project focuses on extracting genres from movie scripts automatically using machine learning techniques.

1.1 Problem Description

In the world of movie production and recommendation systems, accurately classifying movie scripts into their respective genres is crucial for various applications. However, manually categorizing a vast collection of scripts can be time-consuming and prone to human error. Hence, there arises a need for an automated movie script classification system that can efficiently and accurately assign genres.

2 DATA ACQUISITION

We obtained training data through web scraping of two websites, namely Imsdb[3] and Dailyscript[4]. Both of these websites provide a webpage with hypertext links to all available movie scripts on the given website, which is very convenient for web scraping. Another benefit is that the scripts are embedded into the HTML of the webpages. Therefore, there is no need for any complicated parsing, which could have been the case e.g. for PDF files.

In total, we use the scraped data to create 3 data sets. One for each of the respective websites and a third *merged* data set containing deduplicated samples from the *imsdb* and *dailyscript* data sets.

2.1 Imsdb Scraper

The imsdb scraper extracts movie scripts and their respective genres linked from the https://imsdb.com/all-scripts.html webpage. It utilizes the Scrapy[5] library to crawl and scrape the website. Furthermore, we are extending the functionality of the Scrapy spider by specifying the target website, the starting URL and the rules on how to follow links. Then, we are using the beautifulsoup[6] library for extracting genres, the script title and the script itself from the scraped data. The extracted relevant information is later saved into a JSON file.

2.2 Dailyscript Scraper

The dailyscript scraper is designed to scrape movie scripts linked from the https://dailyscript.com/movie.html webpage. It again utilizes the beautifulsoup[6] library for HTML parsing and the Scrapy[5] library for crawling the website. The only major difference is that genres for each script must be additionally scraped from the Imdb[7] website. The extracted data is again saved to a JSON file if the scraping of the genres was successful, otherwise the script is discarded.

3 MODELS

We used several models to perform classification of the movie script genres. One of the models is based on simple statistics, the others are leveraging machine learning techniques.

[3] https://imsdb.com
[4] https://www.dailyscripts.com
[5] https://scrapy.org/
[6] https://beautiful-soup-4.readthedocs.io/en/latest/
[7] https://imdb.com

3.1 Statistical Model

The statistical model is a simplified uni-gram model on the level of words. The simplification lies in not computing probabilities of words, but only counting word occurrences. Furthermore, the prediction is then based on summing the counts of the occurrences rather than multiplying the probabilities, which makes the model robust to unseen words.

We opted for an uni-gram model instead of di-gram, tri-gram and higher N-gram models, since the obtained data sets are relatively small. Utilizing higher N-gram models on these data sets would result in a limited statistical evidence and many unseen combinations later during prediction.

3.2 Machine Learning Models

The machine learning model is more complicated consisting of two sub-models.

First, a pre-trained sentence transformer model leveraging the BERT[8] or GloVe[9] architectures available at SBERT[10] or Hugging Face[11] and in detail described in [2] or [1]. We currently support the following models:

- `all-mpnet-base-v2`,
- `multi-qa-mpnet-base-dot-v1`,
- `all-distilroberta-v1`,
- `all-MiniLM-L12-v2`,
- `average_word_embeddings_glove.6B.300d` (differs from the above models, the sentence embeddings are computed as an average of word embeddings in those sentences generated using the GloVe architecture)

These models are used to generate embeddings of the movie scripts splitted into samples with 256 words each, as this is the upper limit of a sentence length for some of the models.

Second, we use a fully connected neural network for the genre prediction. The network has 2 to 4 hidden layers with the ReLU activation function. All hidden layers apart from the last are followed by a dropout layer with 0.25 dropout probability. The Sigmoid activation function is used as the activation function of the output layer, since the model is trained to perform a multi-class classification. Consequently, we used the binary cross entropy as the loss function. The inputs of this model are the aforementioned embeddings generated by one of the selected sentence transformer models. This model is trained separately on each set of the generated embeddings.

4 METHODS

In this section, we discuss the architecture of our project, how we pre-processed the scraped data, selection of hyper-parameters of the models, how they are trained and how is the prediction of genres performed.

4.1 Architecture

Due to the handling of relatively large files, our pipeline has been designed with multiple scripts that output to files. This approach simulates a stream processing environment, such as Kafka[12], where each file/script is processed sequentially.

[8] Bidirectional Encoder Representations from Transformers
[9] Global Vectors for Word Representation
[10] https://www.sbert.net/docs/pretrained_models.html#sentence-embedding-models/
[11] https://huggingface.co/sentence-transformers/average_word_embeddings_glove.6B.300d
[12] https://kafka.apache.org/

This ensures scalability and enables the use of horizontal scaling tools like Kubernetes[13], by using Docker[14] containers to execute the scripts.

4.2 Pre-processing

The web-scraped data are partial HTML pages still with HTML tags left. Moreover, the scripts itself need cleaning and pre-processing, as they contain special characters, inconsistent capitalization, unnecessary spaces etc. We accomplish the pre-processing mostly using regular expression and natural language processing libraries with the following pipeline:

(1) Cleaning of special characters – this step focuses on removing or replacing specific characters or character sequences present in the script. Whole upper case words are also converted to lower case in this step.
(2) Removal of HTML tags – here, any HTML tags and other markup or formatting found within the scripts are removed.
(3) Removal of punctuation – in this step, we remove punctuation characters in order to not have same words, which differ just by a character, such as comma or dot, at the end. Capital letters following a punctuation character are also converted to lower case in this step.
(4) Removal of special words – this steps removes words, which are there to help the movie crew, when the movie is filmed. These words are e.g. CLOSE ON, CUT and ANGLE ON.
(5) Removal of names – in this step, we remove all words, which still have a starting capital letter, assuming that these words are names of movie characters or places.
(6) Removal of description – here, we remove the script description, which e.g. contains the names of the authors or the date, when the script was published.
(7) Removal of stopwords – this step removes stopwords from the script. We use the nltk[15] library to achieve that. The stopwords removal is applied only for the statistical model, because the sentence transformers already have their additional pre-processing.
(8) Stemming – in this step, stemming is applied to the words of the scripts. For that we again use the nltk[15] library and the step is again skipped for the sentence transformer models.
(9) Genres cleaning – here, we remove unnecessary spaces and special characters from the genres of each script and convert them into lower case letters.
(10) Data set split – in the last step, each data set is splitted to training and test data sets. We chose to have a larger training data sets and earmarked 90 % of the available data for training.

4.3 Statistical Model Training and Prediction

Training of the statistical model is based on counting occurrences of words for each genre. The prediction is then simply performed by looking up and summing the learned counts of occurrences of words appearing in the predicted sample for all genres. The most probable genres are selected as those with the largest sums.

We tried normalizing the counts of occurrences of words in each genre to counter the uneven distribution of genres in the obtained data sets. However, it did not lead to a better performance on the validation data sets. Meaning that the prior probability of genres is significant for correct predictions.

[13] https://kubernetes.io/de/
[14] https://www.docker.com/
[15] https://www.nltk.org/

4.4 Machine Learning Models Training and Prediction

We were not able to fine-tune any of the pre-trained sentence transformer models, due to a lack of available compute resources. However, it is important to mention that fine-tunig these models, ideally with the fully connected neural network already incorporated, for a specific task should improve the prediction performance.

Therefore, the fully connected neural network is trained separately on each set of the generated embeddings with the Adam optimizer using default parameters specified by the TenserFlow[16] library, i.e. 0.001 learning rate, 0.9 β_1 and 0.999 β_2. More precisely, we perform a 2 stage training. The model was first trained using a validation data set earmarked from the training data set with an early stopping maximizing the validation accuracy with patience for 5 epochs as the stopping criterion. Then, the weights and biases were restored and the model was trained again on the whole training data set. The stopping criterion for the second stage was selected as the epoch, where the first training run reached the best accuracy.

Additionally, we used the validation data set to select the best performing number of hidden layers. The model with 4 hidden layers constantly outperformed the smaller models on a majority of the generated sets of embeddings and therefore was selected.

The prediction is performed by pre-processing the given movie script, splitting it into samples with 256 words each, generating embeddings for the samples and predicting each sample. Finally, the predicted probabilities are averaged and the most probable genres are selected.

5 EVALUATION

We evaluated the models using 4 common metrics, namely precision, recall, intersection over union (IoU) and the harmonic mean of precision and recall also known as F1 score. We did not use accuracy, as it is not in general applicable on multi-class classification.

The machine learning models output probabilities of each genre being assigned to a given movie script. Therefore, we considered genres with probability higher than 0.5 as the predicted. The statistical model does not output probabilities. Consequently, we considered the best N predicted genres, where N is the number of genres in the ground truth label of a given movie script.

6 EXPERIMENTS AND RESULTS

We performed 3 experiments to evaluate the whole classification system.

First, we trained all the models on all genres using all the data sets. Furthermore, we evaluated e.g. a model trained on the *imsdb* training data set on all of the available test data sets. However, it must be taken into account, that the data set splits were performed separately, therefore any test data set not intended for a given training data set may contain movie scripts, which the model was already exposed to during training. We believe that this is also the reason, why e.g. some of the models trained on the *dailyscript* training data set perform so well on the *merged* test data set. The evaluation results can be seen in Figures 1, 2, 3 and 4 for the precision, recall, IoU and the F1 score metrics respectively. The results show that the statistical model is a very strong baseline matching and in some cases even outperforming the more advanced models. Another noteworthy observation is that models utilizing the GloVe architecture for generation of embeddings demonstrate comparable performance to the models using embeddings generated by the BERT-based architectures.

[16] https://www.tensorflow.org

Second, we performed a simple ablation study, where the models were trained only on 10 selected genres out of the 23 available. The genres were not selected at random focusing on more common genres with smaller potential for overlap. The selected genres were action, adventure, comedy, crime, documentary, drama, horror, romance, sci-fi and sport. However, the training was performed on all the available scripts, even if a given script was not labeled with any of the selected genre. The evaluation results are shown in Figures 5, 6, 7 and 8 for the aforementioned metrics respectively. The figures show an expected behaviour regarding recall, i.e. being slightly worse, and precision, i.e. being slightly better, compared to the models trained on all the genres. Notably, the overall performance seems to marginally improve as suggested by the F1 score.

Third, we tried to use the trained models to perform a very different task of sentiment analysis. We wanted to analyze if there is a potential for pre-training sentiment analysis models on the task of movie genre prediction. Consequently, we were looking for patterns in the genres predicted for different sentiments, e.g. texts with positive sentiment would be more often predicted as the comedy genre. Precisely, we predicted 1 000 samples of the positive, negative and neutral sentiments each from the Twitter US Airline Sentiment[17] data set available at Kaggle[18]. The predictions of two most probable genres for each sample by the `all-mpnet-base-v2` and `multi-qa-mpnet-base-dot-v1` models trained on the *merged* training data set are summarized in Figures 9 and 10 for the models respectively. The drama genre is the most often predicted followed by the comedy and thriller genres. We can also notice some subtle patterns, e.g the comedy genre is more often predicted for texts of the positive sentiment, while negative sentiment texts are more often assigned with the thriller genre. Overall, considering the results, we conclude that pre-training sentiment analysis models on the task of movie genre prediction could potentially lead only to a minor improvement of such models.

7 GRAPHICAL USER INTERFACE (GUI)

We chose to use the Dash[19] library for an implementation of the GUI, which builds an HTML website. Having a website is very convenient, because it can be hosted and consequently easily accessed via the interned by nearly everyone. The GUI application is build on so called callbacks that make it interactive for the user and at the same time the application can leverage all the files and scripts created in the previous steps. The GUI application is organized in the following four tabs:

- Data Sets – the users can perform web scraping of the two aforementioned websites followed by pre-processing and embeddings generation in this tab. The tab is captured in Figure 11.
- Training – here, the users can train a selected model on a given dataset to predict certain genres. It is also possible to retrain already trained models in this tab. The training tab is shown in Figure 12.
- Prediction – the prediction tab enables the users to predict specified number of genres from an inputted movie script using already trained models. The visual appearance of this tab is captured in Figure 13.
- Evaluation – the last tab provides an interface, which the users can use to evaluate and compare the trained models. We used the Plotly[20] library to provide a simple visual comparison of the models using bar plots. The evaluation tab is displayed in Figure 14.

[17] https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment
[18] https://www.kaggle.com
[19] https://plotly.com/dash/
[20] https://plotly.com

8 CONCLUSION

This project focused on developing a genre classification system using web crawling and machine learning techniques to automatically extract genres from movie scripts. The evaluation of the models was conducted using precision, recall, IoU and the F1 score metrics. A statistical model, which is based on simple word counting, was used as a baseline to validate performance of the more advanced machine learning models. Those were models based on sentence transformers leveraging the BERT or GloVe architectures. The results showed promising performance, where most of the machine learning models outperformed the baseline model, although not by a large margin. However, the margin could be increased by fine-tuning the sentence transformers for this specific task. A simple ablation study demonstrated a potential for another improvement by focusing on specific subsets of genres. Lastly, an experiment with sentiment analysis showed a distant possibility of using pre-training on the task of genre prediction for an improvement of sentiment analysis models.

A FIGURES

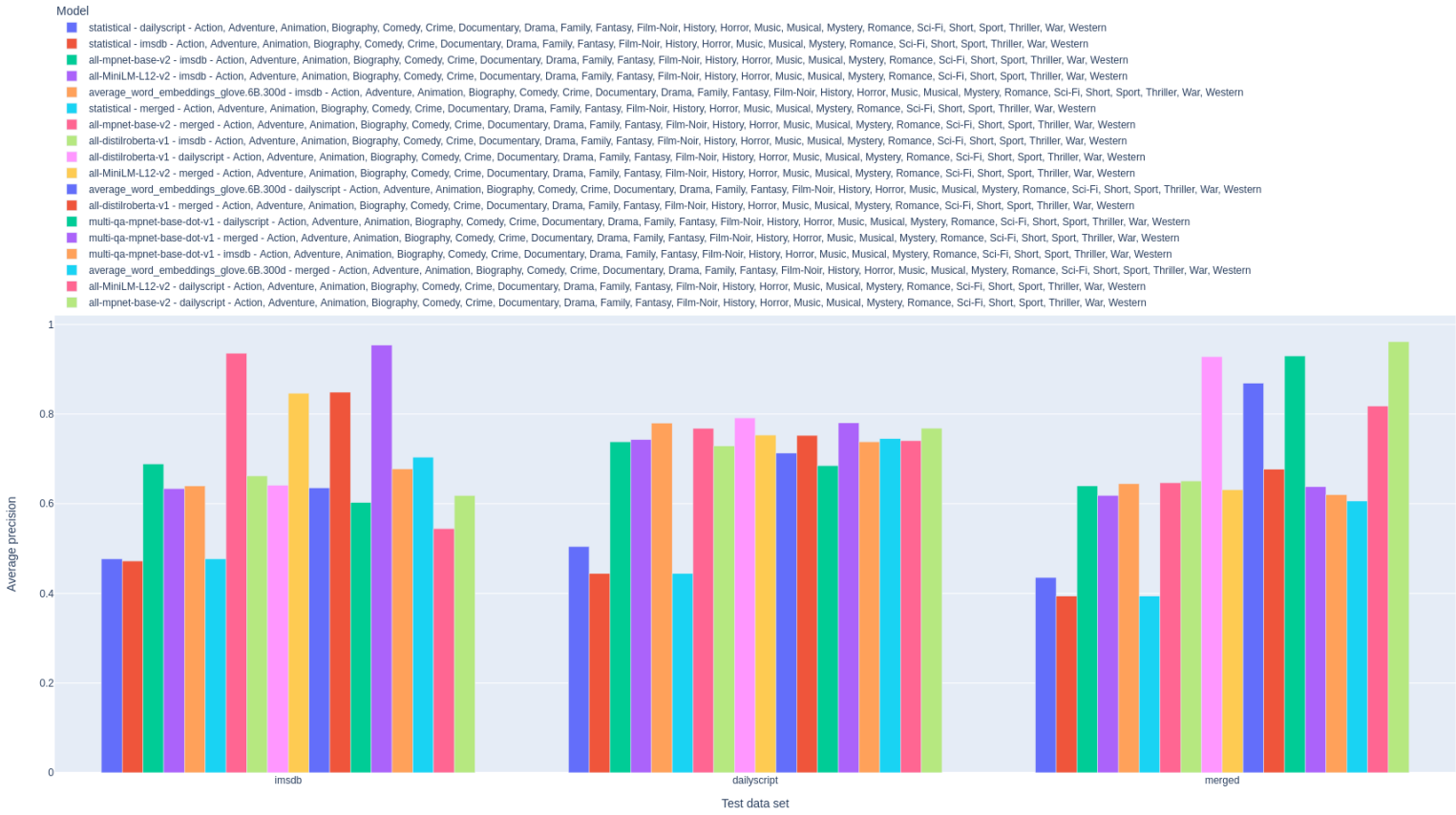

Fig. 1. Average precision of models trained on all genres

Fig. 2. Average recall of models trained on all genres

Fig. 3. Average intersection over union of models trained on all genres

Fig. 4. Average F1 score of models trained on all genres

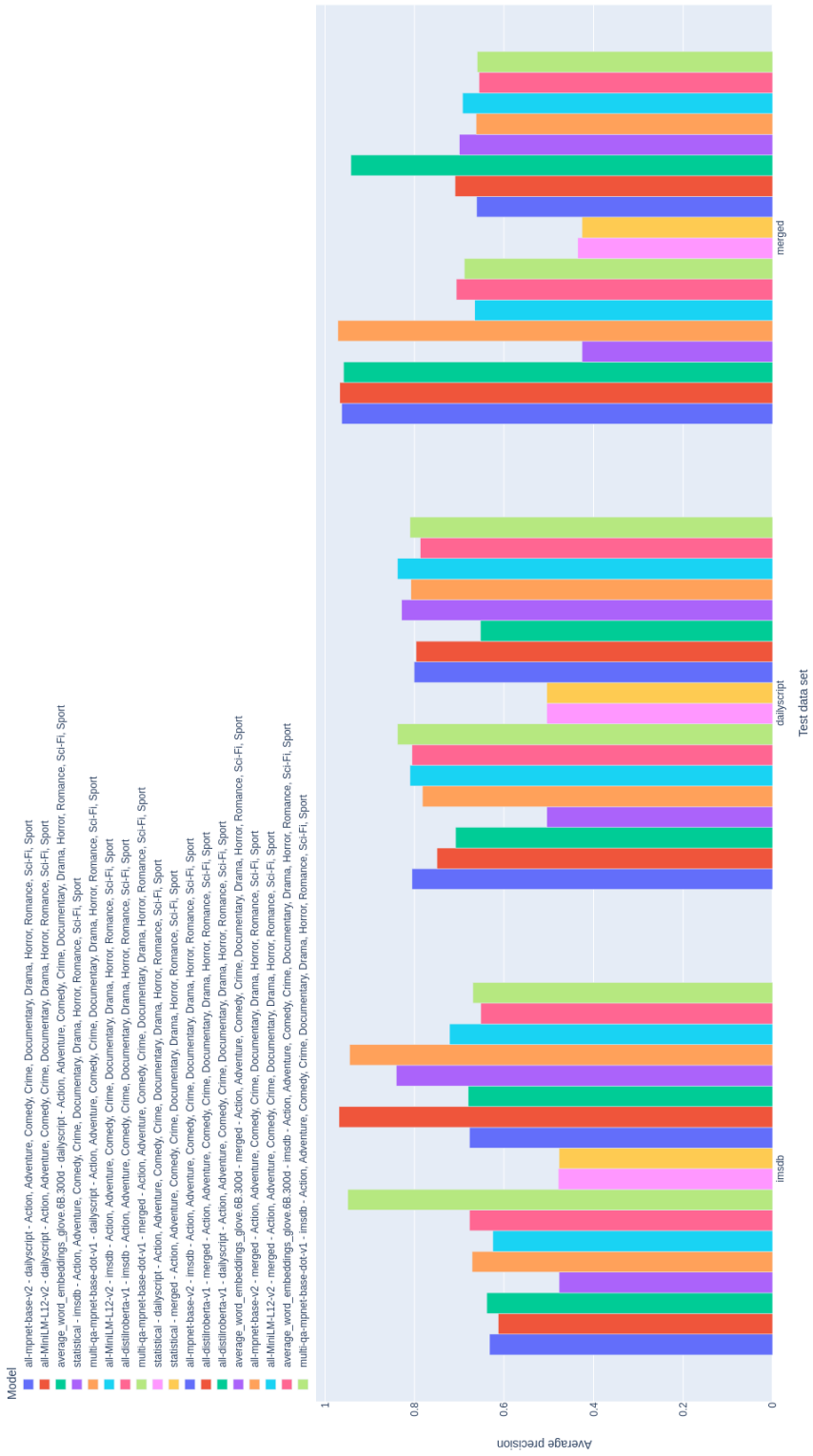

Fig. 5. Average precision of models trained on 10 selected genres

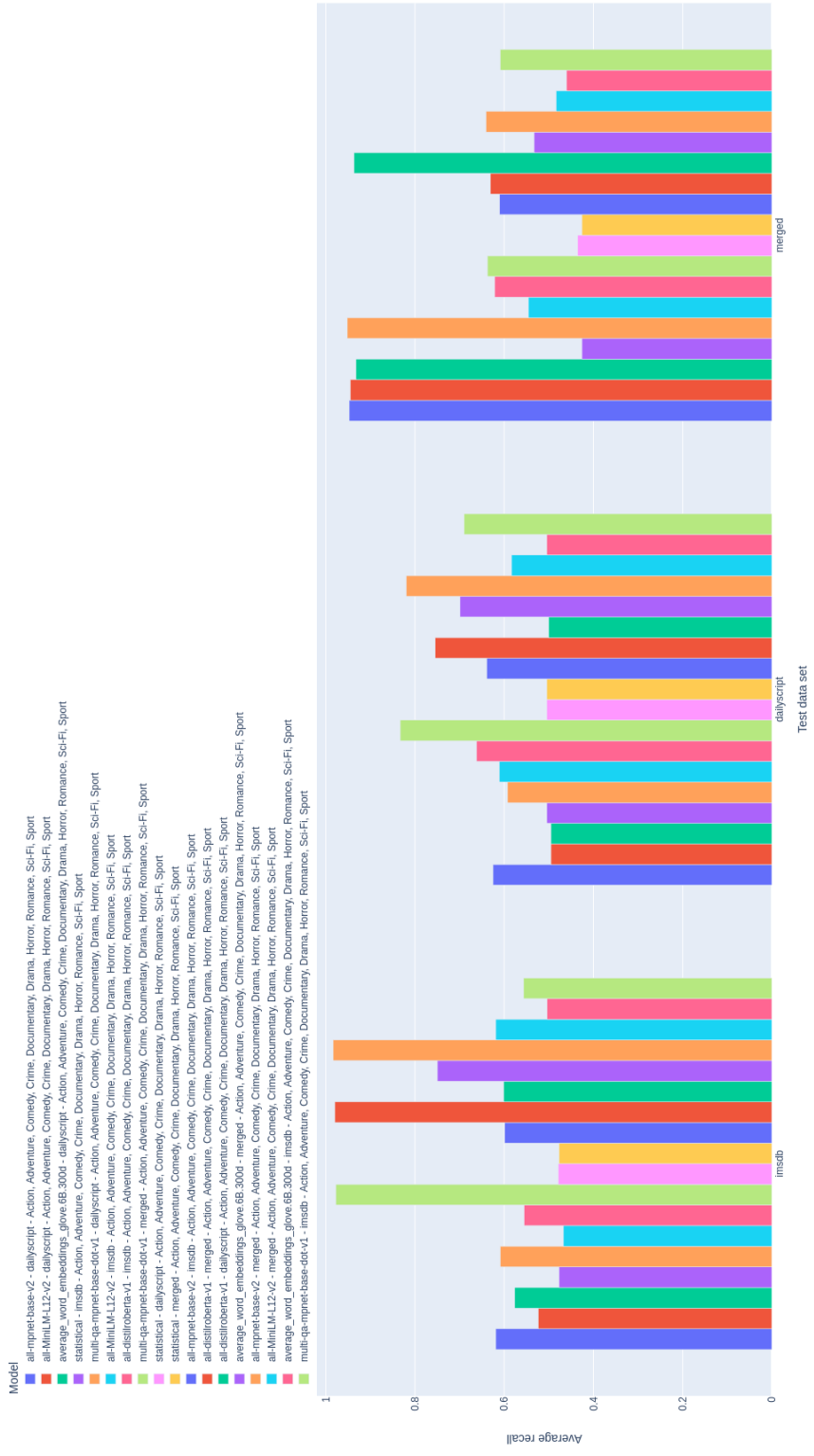

Fig. 6. Average recall of models trained on 10 selected genres

Fig. 7. Average intersection over union of models trained on 10 selected genres

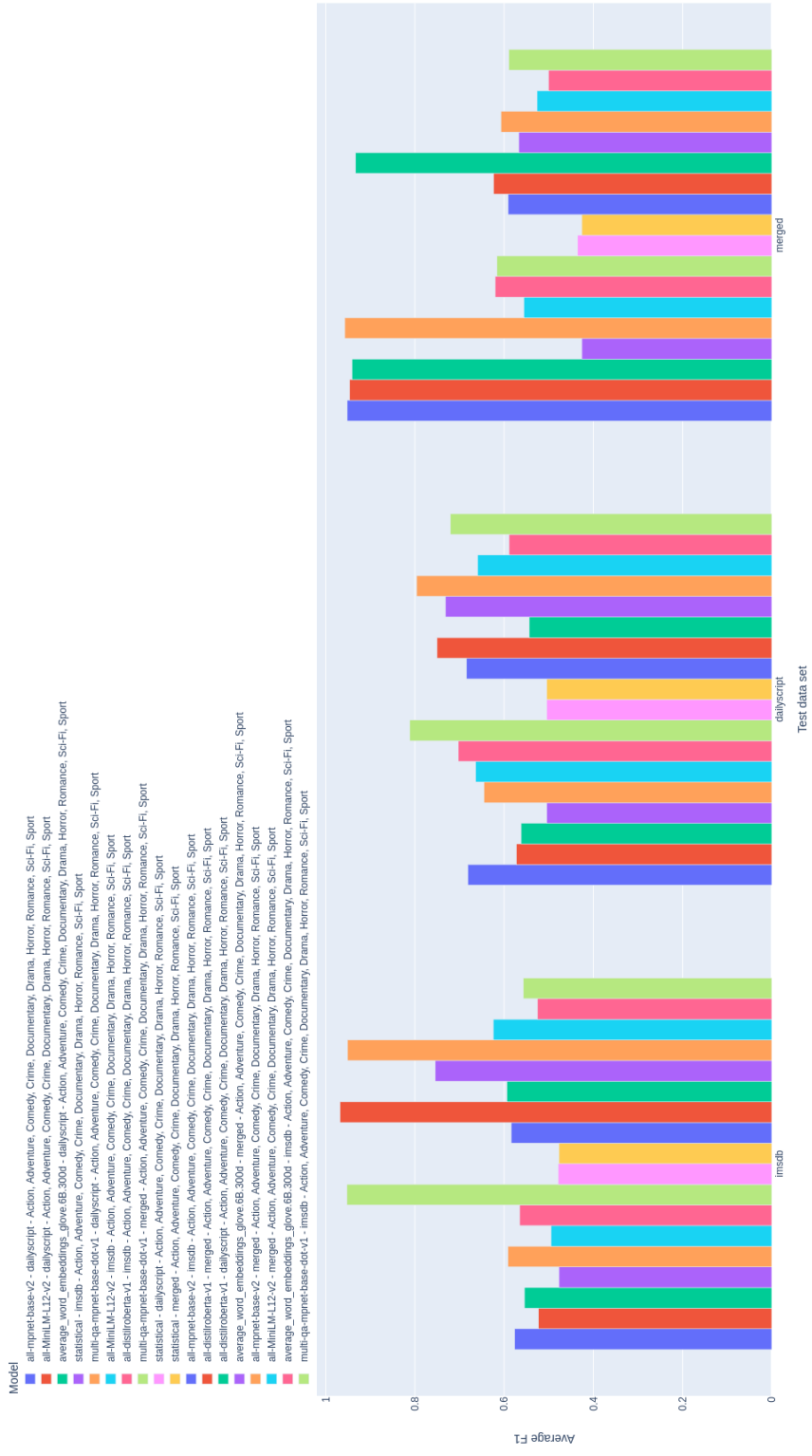

Fig. 8. Average F1 score of models trained on 10 selected genres

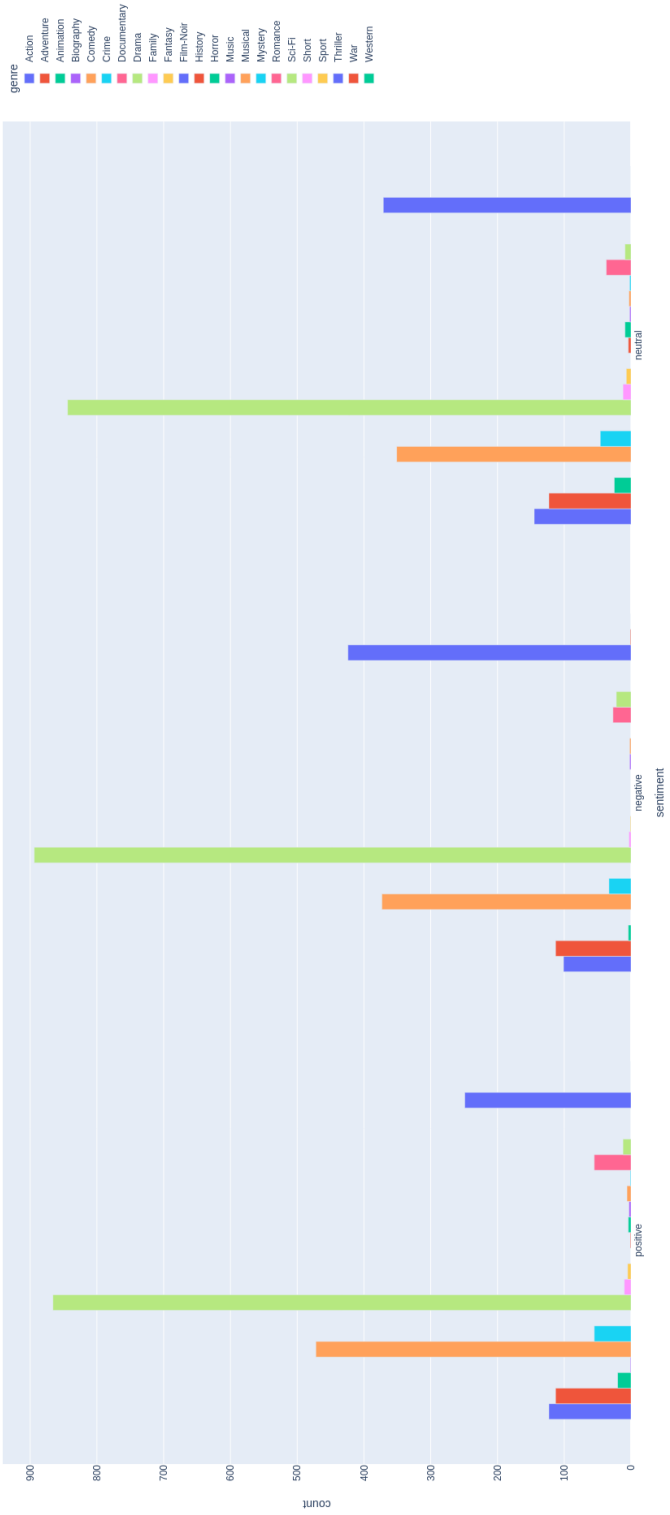

Fig. 9. Genres predicted by the `all-mpnet-base-v2` model trained on the *merged* training data set for texts of different sentiment

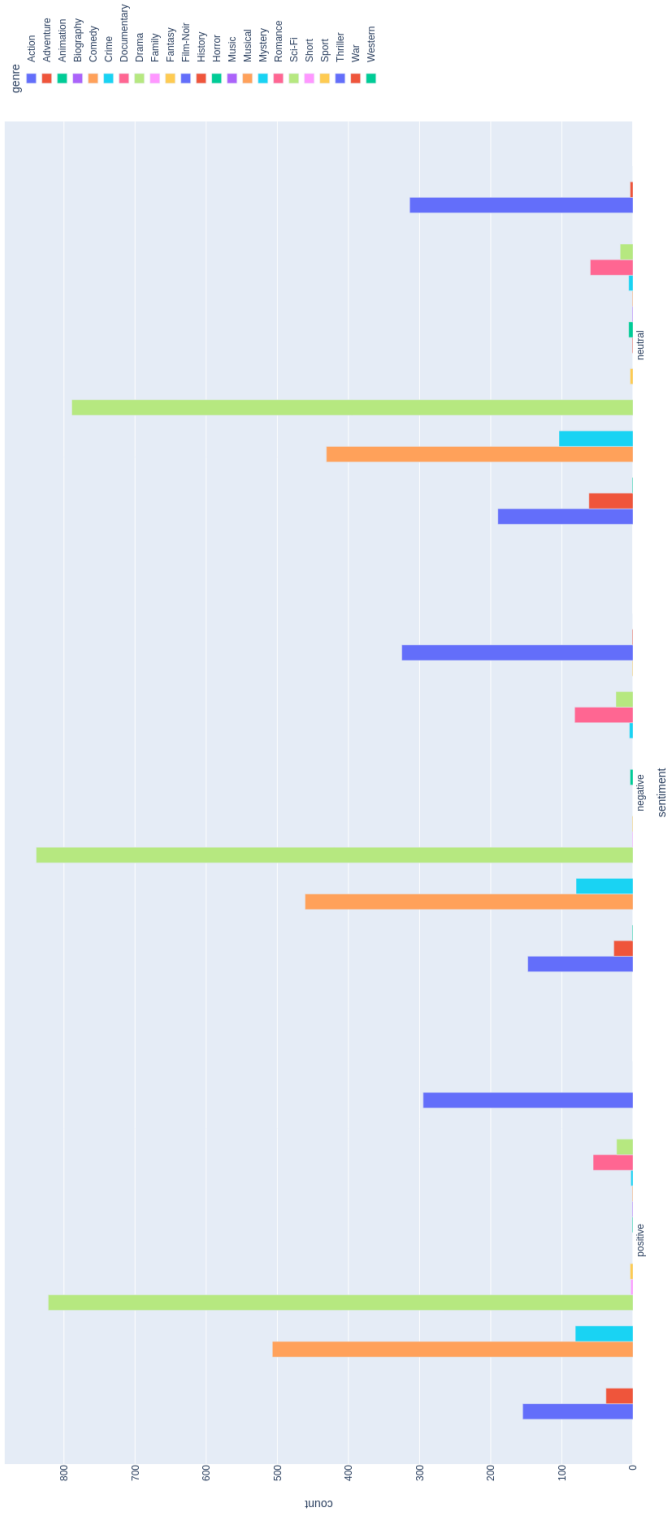

Fig. 10. Genres predicted by the `all-mpnet-base-v2` model trained on the *merged* training data set for texts of different sentiment

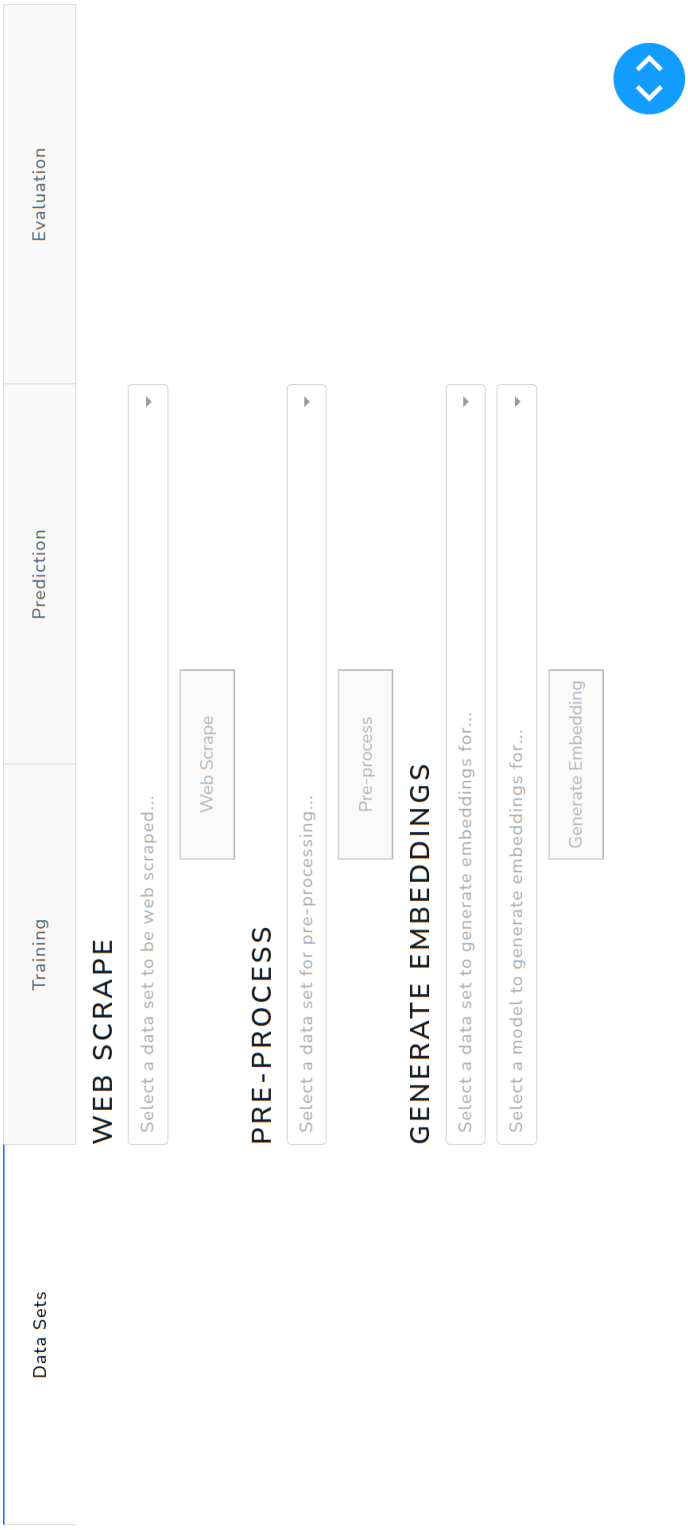

Fig. 11. GUI application – Data Set tab

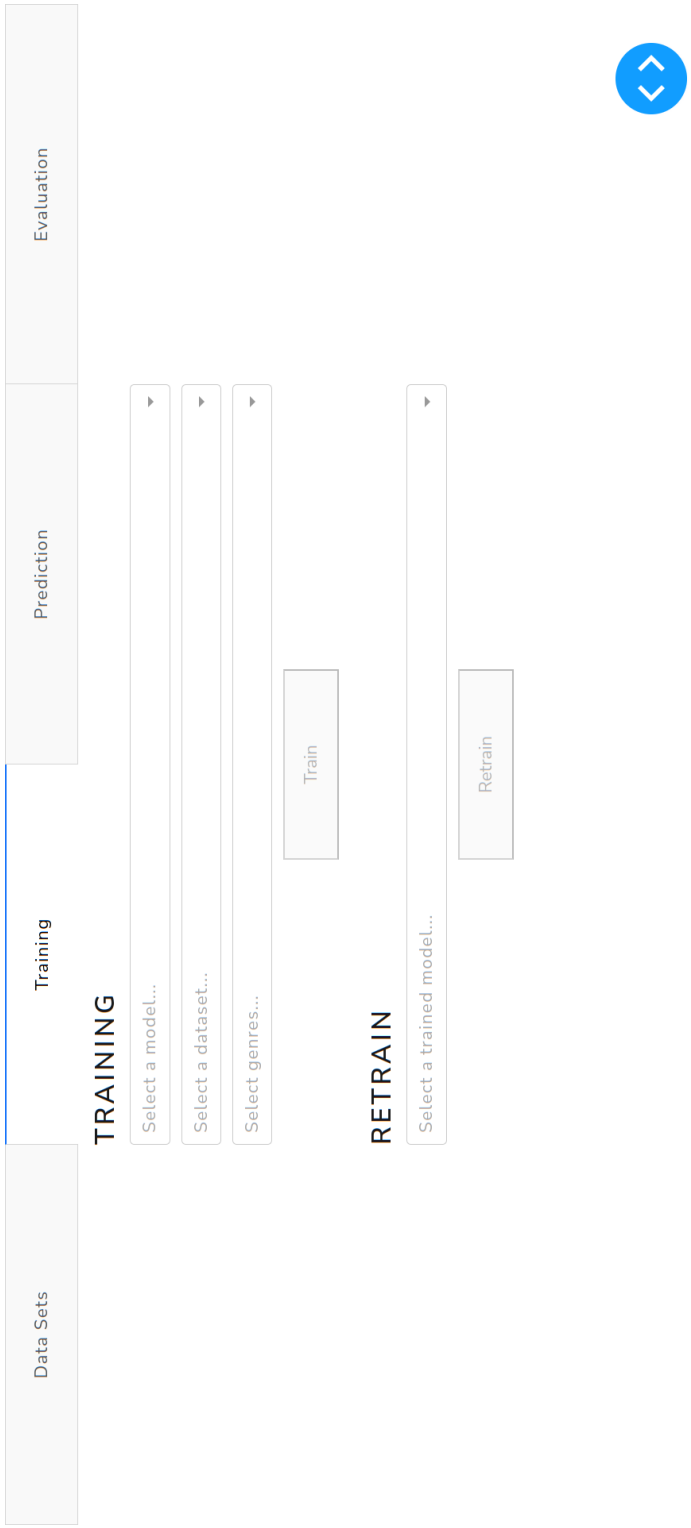

Fig. 12. GUI application – Training tab

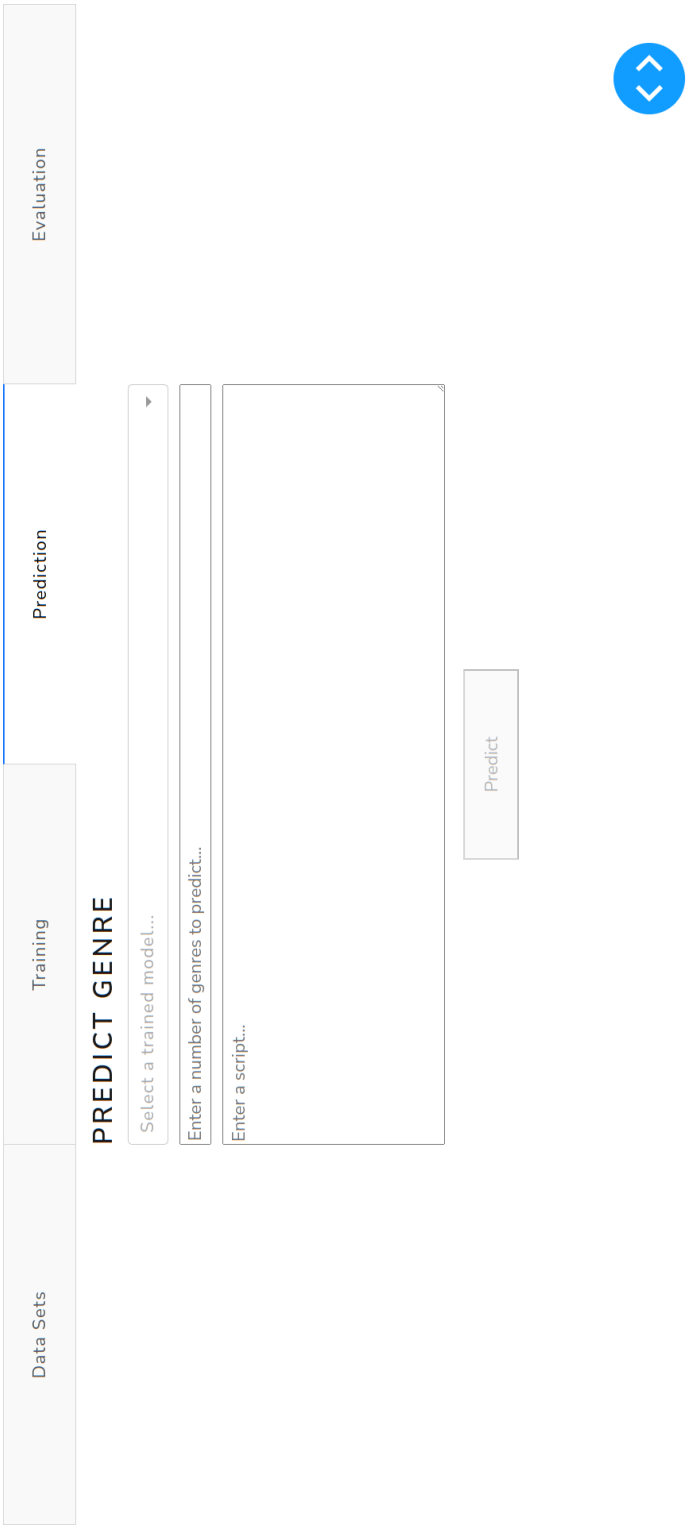

Fig. 13. GUI application – Prediction tab

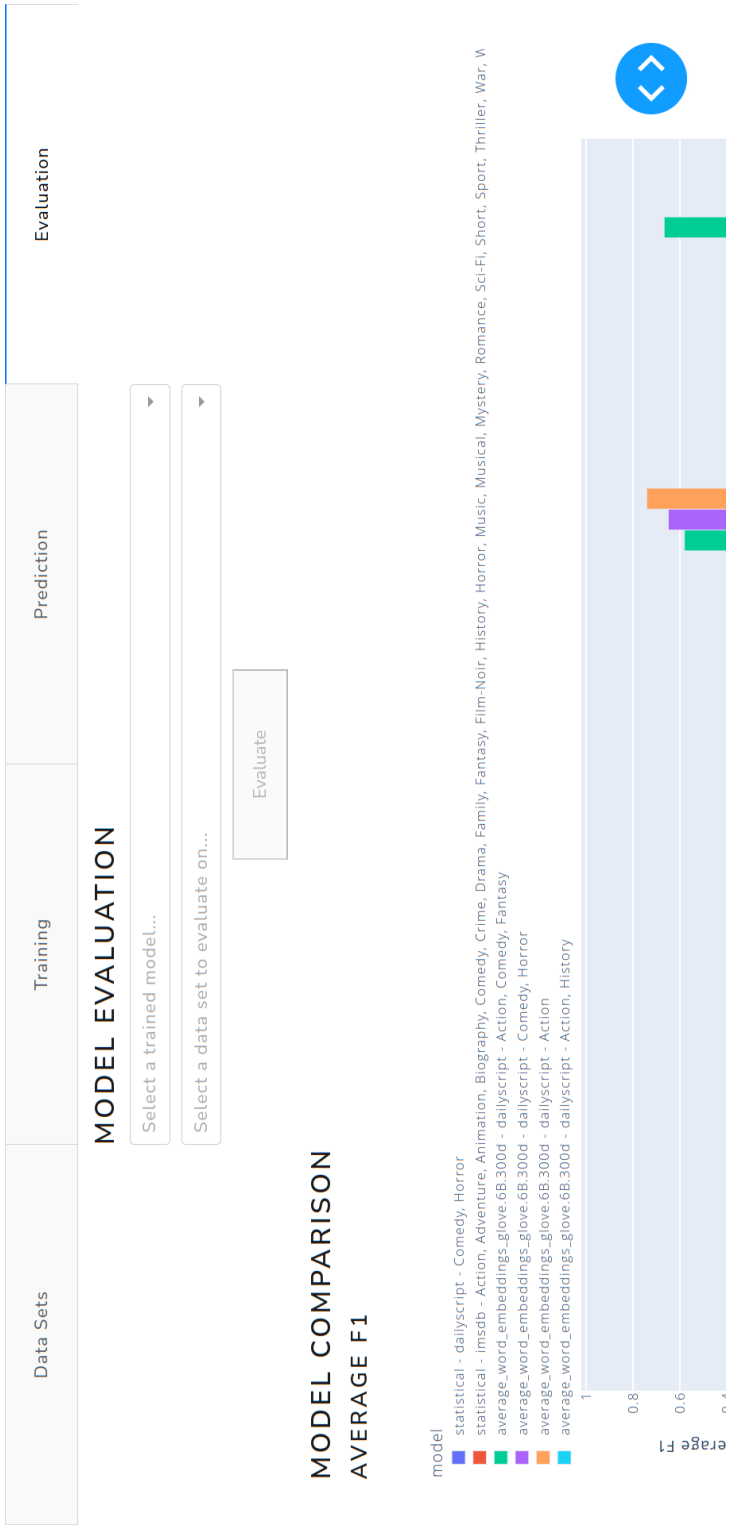

Fig. 14. GUI application – Evaluation tab

B WORK DISTRIBUTION AMONG TEAM MEMBERS

Name	Realized tasks
David Mihola (12211951)	pre-processing, training, prediction, evaluation, GUI application, experiments, report
Johannes Pesenhofer (11771884)	web-scraping, pre-processing, GUI application, report

Table 1. Distribution of work among team members

REFERENCES

[1] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.