Predicting Movie Genres using NLP

João Costa, ist199985 Pedro Miguel, ist112575 Vasco Marques, ist1112489

Professor: Luísa Coheur **Course**: Natural Language - Instituto Superior Técnico **Group:** 30

I. Introduction

The automation of movie genre classification plays a critical role in enhancing search and recommendation systems on streaming platforms. This study applies Natural Language Processing (NLP) techniques to categorize movies into nine genres using a labeled dataset. A combination of traditional machine learning models, such as Support Vector Machines (SVM) with feature extraction techniques like term frequency-inverse document frquency (TF-IDF), and modern deep learning approaches, including transformer-based models like Sentence Transformers and DistilBERT, are explored. The following sections detail the dataset, preprocessing steps, models, and results, highlighting the strengths of different approaches in genre prediction.

II. Models

A. Preprocessing

The dataset, containing 8,041 rows, includes features such as *movie title*, *country of origin*, *genre*, *author name*, and *plot of the movie* was analyzed. Fig. 1 shows that the genre distribution is imbalanced, with certain genres being overrepresented. Further analysis focused exclusively on movie plots, which are highly variable and informative, as discussed in Section II-B.

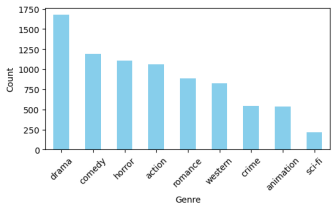

Fig. 1: Histogram of genre counts in the dataset.

The average plot length was 370 words, with a complete word count histogram shown in Fig. 2. An initial word cloud of plot texts (Fig. 3) revealed frequent stop words, punctuation, and citations, which will be addressed during preprocessing.

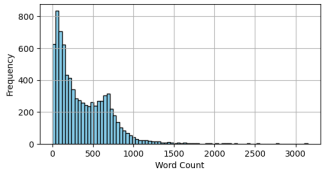

Fig. 2: Histogram of word counts in the movie plots.

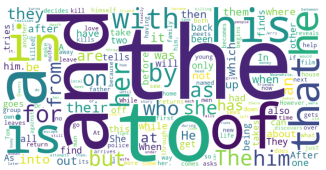

Fig. 3: Word Cloud of movie plot text before preprocessing.

During the initial preprocessing stage, data was cleaned and organized using several techniques, including expanding contractions, standardizing dates, converting text to lowercase, removing punctuation and stop words, lemmatizing with SpaCy, and eliminating references using Regex [1] [2].

After preprocessing, the average plot lengths decreased to 205 words (Figure 4). A new word cloud (Figure 5) highlighted shifts in frequent words, reflecting the effectiveness of the preprocessing steps.

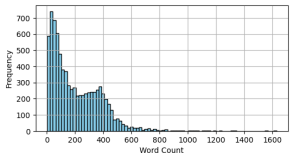

Fig. 4: Histogram of word counts in the movie plots after preprocessing.

Fig. 5: Word Cloud of movie plot text after preprocessing.

B. Models

Various machine learning models and embedding techniques were evaluated, including neural networks, Naive Bayes, SVMs, and boosting methods, combined with vectorization approaches like TF-IDF, Word2Vec, Wiki2Vec as well as advanced embeddings such as Sentence Transformers and DistilBERT. Although FastText and RoBERTa embeddings were considered, limited computational resources constrained their exploration. Table I summarizes the models, and key characteristics. Table II summarizes the different embedding methods used in the experiments. While multiple features were considered, they did not improve accuracy and sometimes reduced it, so the plot was chosen as the primary feature.

TABLE I: Descriptions models characteristics.

Model	Description
MLP	Model non-linear relationships between the plot and genres. Its architecture includes hidden layers activated by ReLU and a softmax output layer for multi-label classification.
Multinomial Naive Bayes	Model discrete features using a multinomial distribution, assuming feature independence and is efficient for text classification.
Gaussian Naive Bayes	Assume a Gaussian distribution for continuous features, predicting genres based on feature means and variances, however less effective for sparse data.
SVM	Employ a linear kernel to find an optimal hyperplane for separating genres.
Unidirectional LSTM	Use an embedding layer to process input sequences, followed by an LSTM layer to capture long-term dependencies.
Bidirectional LSTM	Process sequences in both forward and backward directions using a Bidirectional wrapper around the LSTM layer.
XGBoost	An ensemble method using gradient-boosted decision trees to model complex feature-genre relationships.

III. Experimental Setup and Results

The dataset was split 80/20 for training and testing, followed by tokenization with a 256-token limit per movie plot, chosen empirically to balance computational constraints and data consistency. Results shown in Table III.

Early experiments yielded low accuracy, largely due to class imbalance. While Synthetic Minority Over-sampling Technique (SMOTE) and class weighting were tested to address

TABLE II: Descriptions embeddings characteristics.

Embedding	Description
TF-IDF	Represent text as a sparse vector, highlighting important words relative to the entire dataset.
Word2Vec	Trained on the dataset, this method creates dense word embeddings by learning to predict context words and capturing semantic relationships between terms.
Wiki2Vec	Pre-trained on Wikipedia data, this variant of Word2Vec provides word embeddings that generalize across a large corpus of knowledge [3].
Sentence Transformers	Pre-trained sentence-level embeddings by using transformer-based models trained to capture the semantic similarity between sentences [4].
DestillBert	A distilled, smaller version of BERT that generates context-aware embeddings, preserving most of BERT's performance while being more efficient [5].

TABLE III: Model performance on the original dataset.

Model/Embedding	Macro F1-Score	Weighted F1-Score	Accuracy	Macro precision	Macro recall
FFNN TF-IDF	0.53	0.55	0.54	0.61	0.54
MultinomialNB TF-IDF	0.42	0.48	0.51	0.70	0.40
GaussianNB TF-IDF	0.38	0.40	0.40	0.42	0.36
XGBoost W2V	0.58	0.59	0.59	0.62	0.55
SVM TF-IDF	0.64	0.65	0.65	0.69	0.61
Uni-direct. LSTM W2V	0.60	0.61	0.61	0.63	0.59
Bi-direct. LSTM W2V	0.59	0.62	0.62	0.60	0.60
XGBoost W2V	0.63	0.64	0.64	0.68	0.61
SVM W2V	0.64	0.65	0.65	0.68	0.61
Uni-direct. LSTM Wiki2Vec	0.62	0.63	0.63	0.64	0.61
Bi-direct LSTM Wiki2Vec	0.63	0.65	0.65	0.65	0.63
XGBoost Wiki2Vec	0.62	0.63	0.63	0.66	0.59
SVM Wiki2Vec	0.67	0.66	0.67	0.69	0.64
SVM Sentence Transformers	0.61	0.62	0.62	0.63	0.60
SVM DistillBert	0.64	0.65	0.65	0.65	0.62

this, they resulted in lower accuracy. To address this, the dataset was manually adjusted by reducing entries from the largest class, Drama, and applying data augmentation techniques. Using the Googletrans Python library, back-translation proved particularly effective for augmenting underrepresented genres [6]. This led to a more balanced genre distribution shown in Fig.6. With the revised dataset, the top models were retrained, and results are presented in Table IV.

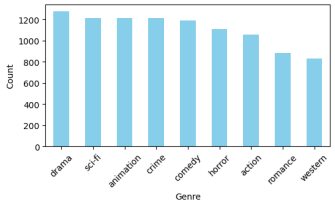

Fig. 6: Histogram of genre counts with new dataset.

TABLE IV: Model performance on balanced dataset.

Model/Embedding	Macro F1-Score	Weighted F1-Score	Accuracy	Macro precision	Macro recall
SVM TF-IDF	0.72	0.71	0.71	0.72	0.71
SVM W2V	0.72	0.72	0.72	0.73	0.71
Bi-direct LSTM Wiki2Vec	0.71	0.70	0.69	0.71	0.71
SVM Wiki2Vec	0.72	0.72	0.72	0.73	0.72
SVM DistillBert	0.72	0.72	0.72	0.72	0.72

The best models, SVM with Wiki2Vec and SVM with DistillBert, were selected for detailed accuracy analysis by genre, as seen in Table V. Also, the confusion matrices for these two models are in Fig 7 and Fig 8, respectively.

IV. Discussion

This study evaluates machine learning techniques for automating movie genre classification, highlighting the importance of preprocessing, feature selection, and model choice. Both traditional methods and advanced deep learning approaches yielded high accuracy (approximately 70%). Although moving from TF-IDF to modern embedding methods like DistilBERT improved performance, the gains were less significant than anticipated, likely due to genre overlap.

TABLE V: Accuracy per movie genre.

Model/Embedding	SVM Wiki2Vec	SVM DistillBert
Drama	0.52	0.52
Sci-fi	0.79	0.88
Animation	0.92	0.89
Crime	0.68	0.73
Comedy	0.59	0.55
Horror	0.84	0.78
Action	0.67	0.62
Romance	0.64	0.56
Western	0.92	0.92

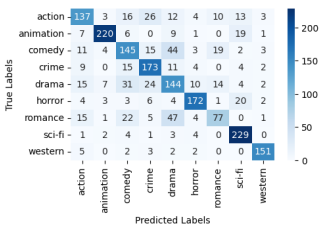

Fig. 7: Confusion matrix of SVM with Wiki2Vec.

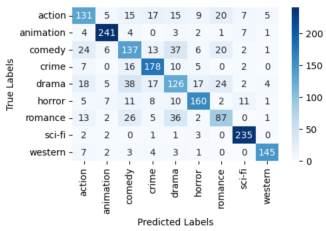

Fig. 8: Confusion matrix of SVM with DestillBert.

Genre overlap poses challenges in classification because many movies blend elements from different genres and defy rigid genre categorizations. For instance, *"Three Strangers"* features clear crime elements like robbery and murder but was predicted as drama, likely due to its focus on emotional conflicts. Similarly, *"Return of Mr. Superman"* contains clear action elements, yet its superhero's unconventional outfit ("heavy sweatshirt, skull cap, and goggles") resulted in misclassification as comedy.

These examples illustrate the challenges faced by our models, with comedies and dramas achieving only 59% and 52% accuracy, respectively, while animated films and Westerns excelled with 92% due to their distinct traits. This difficulty in classification is not limited to machine learning models; it's worth noting that genre classification is also a challenging task for experienced viewers.

V. Future Work

Future research directions include investigating the impact of incorporating additional text data rather than relying solely on data augmentation through back-translation, efforts will be directed towards collecting real movie data from underrepresented genres to improve dataset diversity.

With an expanded dataset, experiments will explore training separate models for each genre, allowing for specialization and potentially better performance.

Additionally, alternatives like GPT-2 or GPT-4 API will be investigated for fine-tuning in genre classification [7], which could improve model accuracy and understanding of genres.

While this study is primarily focused on genre classification, a potential application of this work is in movie recommendations. Integrating genre classification with user preferences could lead to more personalized and relevant content suggestions, enhancing the overall user experience.

REFERENCES

[1] P. v. Kooten, "contractions," *PyPI*. [Online]. Available: https://pypi.org/project/contractions/

[2] Explosion., "Industrial-strength natural language processing (nlp) in python," *PyPI*. [Online]. Available: https://pypi.org/project/spacy/

[3] S. Ousia, "Wikipedia2vec," acessed on 2024-10-16. [Online]. Available: https://wikipedia2vec.github.io/wikipedia2vec/

[4] T. H. F. team, "Sentense transformers," acessed on 2024-10-16. [Online]. Available: https://huggingface.co/sentence-transformers

[5] ——, "Destill bert," acessed on 2024-10-16. [Online]. Available: https://huggingface.co/docs/transformers/en/model_doc/distilbert

[6] Ssut., "Free google translate api for python. translates totally free of charge." *PyPI*. [Online]. Available: https://pypi.org/project/googletrans/

[7] T. H. F. team., "State-of-the-art machine learning for jax, pytorch and tensorflow," *PyPi*. [Online]. Available: https://pypi.org/project/transformers/