Natural Language Project Report

Natural Language 2024/25 | Group 3

Gonçalo Carvalho
ist199227

Leonor Carvalho
ist199623

Guilherme Leitão
ist199951

1 Models

In order to enhance the dataset for training, the following preprocessing steps were carried out in sequence:

(1) **Concatenation of text fields**: Different combinations of the text fields were tested, concatenating them into a single feature to provide a comprehensive representation of the data; the best results were obtained by combining the *from*, *title* and *director* with the *plot* field.
(2) **Synonym-based data augmentation**: To address class imbalance, we used the NLPAug library [1] to perform data augmentation through synonym replacement, substituting words in sentences from minority classes with their synonyms.
(3) **Data normalization**: We replaced missing values with empty strings, removed special characters and digits, and converted all text to lowercase.
(4) **Expansion of Contractions**: Contractions in short texts were expanded to their full forms for improved clarity.
(5) **Part-Of-Speech Tagging and Lemmatization**: The Python library NLTK [2] and the WordNet® database [3] were used for POS tagging, followed by lemmatization. This ensures that words are reduced to their base forms despite syntactic variations.
(6) **Removal of Stop Words**: Common stop words were removed to focus on the most informative terms. The English stop words list from the NLTK library [2] was used, and a minimum document frequency threshold was defined for additional filtering.

For feature extraction, we used a count vectorizer and a TF-IDF transformer to convert the preprocessed data into numerical vectors. Two models from the sklearn [4] library were used to classify the train set:

- **Support Vector Machine (SVM)**: Discriminative classifier that maximizes the margin between classes.
- **Multinomial Naive Bayes (MNB)**: Probabilistic classifier for text; assumes independent features.

2 Experimental Setup and Results

We used GridSearchCV [5] from sklearn library to systematically evaluate various combinations of hyperparameter values (by cross-validation) and determine the optimal configuration that maximizes accuracy.

- **Dataset**: Stratified split (per genre) *train.txt* into training (90%) and test (10%) sets.
- **Evaluation Measures**: Used accuracy, precision, recall, F1-measure and confusion matrices; calculated per-genre accuracy.
- **Best Model Parameters**: TF-IDF with a maximum of 10000 features, an ngram range between 1 and 5 tokens (found after manual fine-tuning - Fig. 1), sublinear tf scanning and the following sets of stop-words:
  - **English**: we calculated every word's DF score and removed only those with a frequency higher than 15%. Later, as can be seen in Fig. 2, we tested different thresholds to find the optimal value to be 0% (meaning all stop-words were removed).

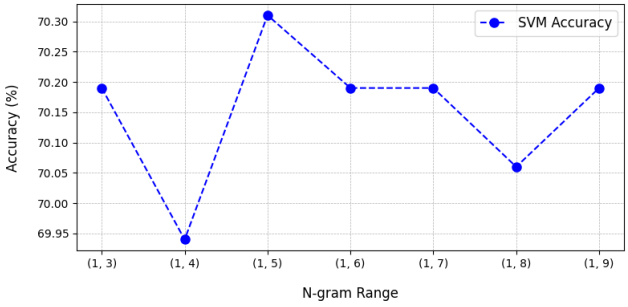

Figure 1: N-gram Range.

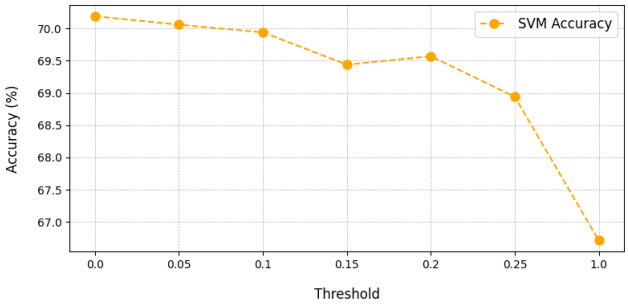

Figure 2: Document Frequency Threshold.

  - **Custom**: even though they are not stop-words by definition, we added specific *genre* words — "action", "animation", "comedy", "crime", "horror" and "western" - to the stop-words list and removed them from the dataset to reduce "*genre* contamination".

For the models, we applied hyperparameter tuning to the following parameters, having achieved the respective optimal values:

- **SVM**: $kernel = sigmoid; C = 1; class\_weight = None$
- **MNB**: $alpha = 0.1; fit\_prior = True$

Table 1 presents the overall best mean cross-validation accuracy.

Table 1: Overall Accuracy per Model.

Model	Accuracy (%)
Support Vector Machine	70.31
Multinomial Naive Bayes	64.97

As we can see, the best model was the SVM, henceforth every analysis will be based on its results.
Table 2 shows the metrics for each genre.

Table 2: Per-Genre Metrics.

Genre	Precision	Recall	F1-Score
Action	0.68	0.65	0.66
Animation	0.86	0.80	0.83
Comedy	0.66	0.64	0.65
Crime	0.53	0.56	0.54
Drama	0.61	0.72	0.66
Horror	0.87	0.83	0.85
Romance	0.57	0.48	0.52
Sci-Fi	0.85	0.52	0.65
Western	0.94	0.98	0.96

Table 3 illustrates the accuracy and F1-score per model with a modified preprocessing hyperparameter.

Table 3: Per-Hyperparameter Metrics.

Metrics	Best Model	No Lemmatize	"Plot" Only	Keeping *genre* Words	Keeping Stopwords	No Augmentation
Accuracy (%)	70.31	69.19	67.70	69.81	66.83	67.58
F1-Score	0.70	0.69	0.68	0.70	0.67	0.67

Fig. 3 shows the confusion matrix for the best model, which represents the optimal combination of hyperparameters from GridSearchCV.

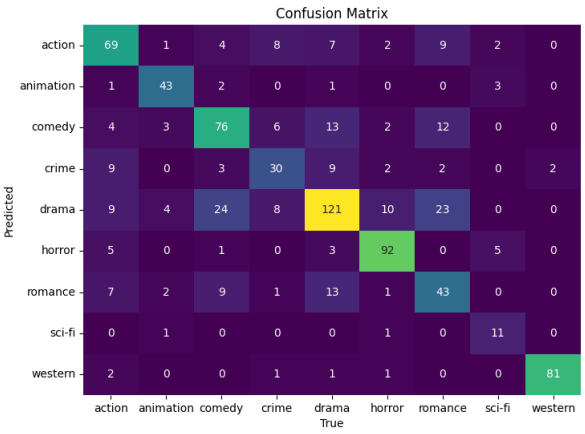

Figure 3: Confusion Matrix for SVM.

3 Discussion

As shown in Fig. 4, most *genre* words appeared with higher frequency in *genres* not matching their own (the effect is scaled with data augmentation). These words, being directly associated with the classification task of the model, can lead the model to predict the class of the word rather than the actual class of the movie. This means that the word may be interpreted outside of its semantic context, leading to inaccurate predictions.

With this in mind, we decided to remove these words from the dataset. However, we decided not to remove "sci-fi" due to tokenization issues related to the hyphen. Additionally, we kept the word "romance" because we noticed a significant drop in precision for this class after

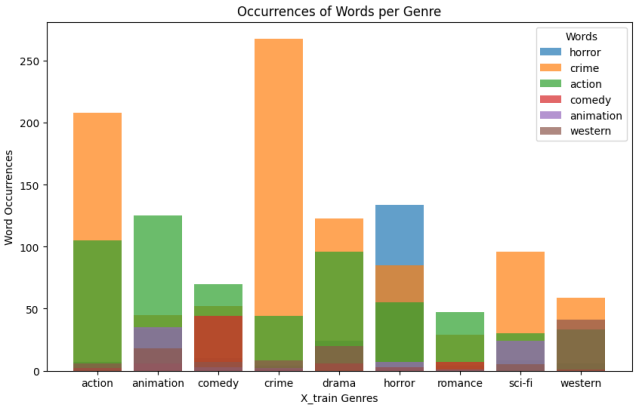

Figure 4: Distribution of *genre* words.

removing it. The model struggled to interpret the concept of a romance film, and retaining the word improved performance. This suggests that "romance" is an essential word for describing this type of movie, as romance movies are generally not so closely associated with other genres. Instead, it is other genres that often incorporate romantic elements, highlighting the uniqueness of romance as a genre.

Including the word "romance" allows the model to more easily recognize the category of romantic films, thereby improving the accuracy of predictions. This experience underscores the importance of considering semantic context and word associations when dealing with text classification in natural language processing.

The SVM model outperformed MNB in overall accuracy, likely due to its effectiveness in high-dimensional spaces like TF-IDF vectors.

Common misclassifications occurred between similar genres:

- **Drama vs. Romance**: We think this common mislabelling be due to the fact that in both genres plots featuring relationships facing challenges are common as well as other common themes.
- **Drama vs. Comedy**: We believe this mislabelling might occur due to the existence of comedic dramas, or dramas in which an unusual or humorous character or situation are present.

Examples of Misclassified Plots

- **Plot**: "Wealthy friends are entangled in a dispute that strains their children's engagement through deceit, love and the revelation of the truth, they reconcile and allow their children to marry." **Actual Genre**: Drama **Predicted**: Romance
- **Plot**: " A virginal sixth-former delivering groceries for the local supermarket, is more interested in seducing the local girls." **Actual Genre**: Drama **Predicted**: Comedy

4 Future Work

- **Hyperparameter tuning**: adjusting the number of synonyms used for data augmentation and fine-tuning the amount of generated data
- **Data augmentation techniques**: paraphrasing (back translation, text expansion and entity substitution) or GANs
- **Latent Dirichlet Allocation** to identify underlying themes or topics present in the text data.
- **Deep Learning Models**: LSTMs or Transformers for better semantic understanding.

References

[1] E. Ma, "Nlp augmentation," https://github.com/makcedward/nlpaug, 2019.

[2] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[3] G. A. Miller, "WordNet: A lexical database for English," in *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. [Online]. Available: https://aclanthology.org/H94-1111

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.