Final Report

Luca Serfilippi, Sara Furlani, Andrea Tribotti

5 September 2025

Abstract

In this work, we present the development of three machine learning models designed to predict six predefined tags from a corpus of stories. We begin by discussing the modeling choices and the adopted training procedure. Subsequently, we compare the three approaches and provide an in-depth analysis of the results obtained.

1 Introduction

The work on this project was divided among the team members: Sara Furlani and Luca Serfilippi focused on the development of the first two models, including dataset preparation and vocabulary construction when necessary, while Andrea Tribotti was responsible for the implementation and fine-tuning of the pre-trained DistilBERT model.
Building on the problem statement summarized in the abstract, this section introduces the scope and objectives of the project. The project aims not only to implement three machine learning models to perform multi-label classification, but also to compare their performance and highlight their strengths and limitations. The report is organized as follows: Section 2 describes the models and training setup. Section 3 discusses the results obtained, provides a comparative analysis, and presents the conclusions.

2 Models and Trainig Setup

2.1 Linear Model

We reduced the training set to 250,000 stories (used for both training and validation), mainly for computational reasons. Our model relies on two pre-implemented components: TfidfVectorizer and Logistic Regression.

With TfidfVectorizer, we keep the 5,000 most relevant features. TF-IDF represents texts numerically, giving higher weight to words that are frequent in a document but less common across the entire dataset, while downweighting very common terms (e.g., articles). In addition, we configure the model to consider both unigrams and bigrams, which allows us to capture not only the meaning of individual words but also the contextual information given by word pairs.

We then apply Logistic Regression as the classifier, setting the parameter `class_weight="balanced"` so that the model automatically compensates for class imbalance.

Finally, we wrap Logistic Regression in a One-vs-Rest Classifier, which allows us to extend the approach to multi-label classification.

2.2 Non Linear Model

The first step was to build our own vocabulary. To do so, we randomly sampled 250000 stories from the training set, using them exclusively for vocabulary construction. Our vocabulary includes words, punctuation marks, and numbers.

Next, we tokenized the stories, representing them as vectors of length 256. To ensure that all sequences had the same length, we introduced a special `[PAD]` token: whenever a story was shorter than 256 tokens, we padded it with this token.

We also introduced an `[UNK]` token to handle out-of-vocabulary cases. Whenever a word in the training set did not appear in the stories used to build the vocabulary, it was replaced with this unknown token.

The proposed model is a Transformer-based text classifier. It is composed of the following components:

1. **Token Embedding Layer**: maps each token into a dense vector representation of dimension 128. Padding tokens are handled explicitly.

2. **Learned Positional Embedding**: adds a trainable positional vector to each token embedding, enabling the model to capture word order.

3. **Transformer Encoder**: a single encoder layer with 4 attention heads and a feed-forward network of dimension 128. This captures contextual dependencies across the sequence.

4. **Max Pooling**: it takes the sequence of contextualized embeddings and turns it into a fixed-size representation by picking the maximum value across the sequence.

5. **Output Layer**: a fully connected linear layer that projects the pooled representation into 6 logits, one for each possible label.

Regarding the training of this model, we employed the `BCEWithLogitsLoss` function, which corresponds to the Binary Cross-Entropy loss with logits. Following the multi-label setting, each tag was treated as an independent binary classification problem. The optimization was carried out using the Adam optimizer with a batch size of 128 and a learning rate of 0.001. The model was trained for 5 epochs.

2.3 DistilBERT and Fine-Tuning for Multi-Label Text Classification

DistilBERT (Distilled Bidirectional Encoder Representations from Transformers) is a transformer-based language model developed by Hugging Face, derived from Google's original BERT model through knowledge distillation. While BERT-base consists of 12 transformer encoder layers and about 110 million parameters, DistilBERT reduces this to 6 layers with a hidden size of 768 and 12 self-attention heads, totaling approximately 66 million parameters. Despite being smaller, DistilBERT retains about 97% of BERT's performance on language understanding benchmarks, while being roughly 40% faster during inference.

Like BERT, DistilBERT is pre-trained on large corpora (e.g., Wikipedia and BooksCorpus) using objectives such as Masked Language Modeling. The absence of the Next Sentence Prediction task and its compressed architecture make it more efficient, particularly for applications where computational cost is a concern.

In this project, DistilBERT (uncased version) was fine-tuned for **multi-label text classification**. To adapt the model to this setting, the final classification layer was replaced with a fully connected output layer matching the number of possible tags. The model was trained using **Binary Cross-Entropy with Logits Loss**, which is suitable for multi-label problems, and optimized with the **Adam optimizer**. During training, the dataset was tokenized with the DistilBERT tokenizer, and mini-batches (of size 64) were fed into the model. Fine-tuning was performed for 5 epochs with a small learning rate (0.0001), in order to preserve the pre-trained linguistic knowledge while adapting to the specific task.

3 Results and Conclusions

To evaluate the performance of the models, several classification metrics were considered. In addition to *accuracy*, which measures the percentage of correctly classified labels, we used *precision*, *recall*, and the *F1-score*.

Precision indicates the proportion of correct predictions among all predictions made for a given label, and is therefore useful to assess how reliable the model is when assigning a tag. *Recall*, on the other hand, measures the model's ability to correctly identify all occurrences of a label present in the data, serving as an indicator of sensitivity. The *F1-score* represents the harmonic mean of precision and recall, providing a single value that balances the two metrics, which is particularly useful in cases of class imbalance.

3.1 Test Set Performance

The following tables report the performance of the models on the test set, for each individual label. This presentation allows observing differences in the models' behavior across the various tags. Notably, the labels *Conflict* and *Foreshadowing* consistently achieve lower performance compared to the others, as indicated by their lower F1-scores, precision, and recall values in all models.

In the first model (logistic regression), recall values are generally higher than precision. This indicates that the model tends to be more inclusive in its predictions: it successfully captures most true cases (low false negatives), but at the cost of misclassifying more negatives as positives (high false positives). In other words, the model prioritizes not missing relevant cases over being selective.

Tag	Accuracy	Precision	Recall	F1
BadEnding	96.32%	0.7213	0.9610	0.8241
Conflict	79.73%	0.2845	0.7993	0.4197
Dialogue	89.87%	0.9156	0.8985	0.9070
Foreshadowing	80.36%	0.2961	0.7721	0.4281
MoralValue	95.53%	0.7118	0.9525	0.8148
Twist	92.88%	0.7755	0.9022	0.8340
Average	89.11%	0.6175	0.8809	0.7046

Table 1: Performance of the first model on the test set per tag.

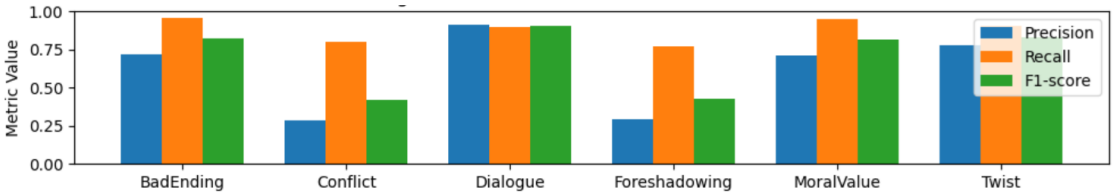

Figure 1: Tag-level Metrics for the First Model (Test Set)

In the second Model (non-linear neural network), the opposite trend is observed: precision is generally higher than recall. This suggests that the model is more conservative when assigning labels, producing fewer false positives but missing a larger portion of true cases (higher false negatives). Consequently, the model prioritizes being accurate when it predicts a class rather than covering all true cases.

Tag	Accuracy	Precision	Recall	F1
BadEnding	97.56%	0.847	0.889	0.867
Conflict	92.19%	0.641	0.337	0.442
Dialogue	90.95%	0.880	0.967	0.922
Foreshadowing	91.84%	0.707	0.204	0.362
MoralValue	96.48%	0.902	0.740	0.813
Twist	94.93%	0.892	0.847	0.869
Average	93.99%	0.811	0.671	0.712

Table 2: Performance of the second model on the test set per tag.

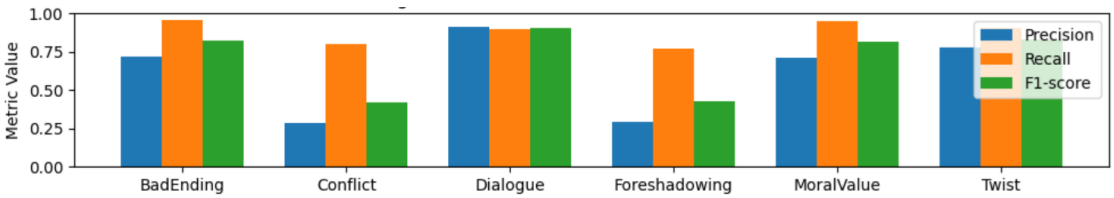

Figure 2: Tag-level Metrics for the Second Model (Test Set)

In the third model (distilBERT), both accuracy and F1-score surpass those of the other approaches. Recall values lie between the two baselines (*non-linear NN* < *distilBERT* < *logistic regression*), while precision is higher than logistic regression but slightly below the non-linear network (*logistic regression* < *distilBERT* < *non-linear NN*). This shows that distilBERT achieves the best overall balance, combining strong coverage with reliable predictions.

Tag	Accuracy	Precision	Recall	F1
BadEnding	98.66%	0.955	0.893	0.923
Conflict	93.31%	0.688	0.495	0.576
Dialogue	92.52%	0.902	0.970	0.934
Foreshadowing	90.82%	0.517	0.553	0.534
MoralValue	97.40%	0.902	0.839	0.869
Twist	93.88%	0.786	0.951	0.860
Average	94.43%	0.791	0.783	0.783

Table 3: Performance of the third model on the test set per tag.

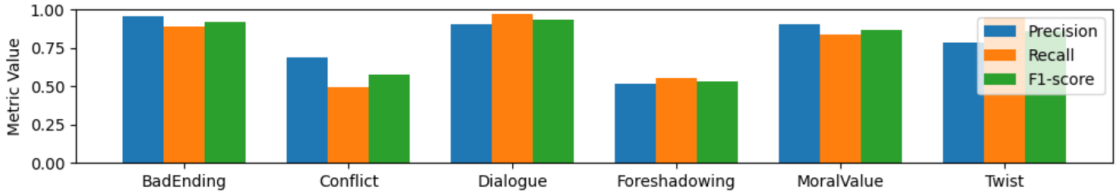

Figure 3: Tag-level Metrics for the Third Model (Test Set)

3.2 Label-Specific Observations

Some labels, like *BadEnding*, are easier to predict even with a basic model such as logistic regression. This is because they are often linked to clear keywords or phrases that strongly indicate the label. In these cases, the model does not need to understand word order or deeper context to make correct predictions.

On the other hand, labels like *Foreshadowing* or *Conflict* are more difficult. They depend on subtle context, how words relate to each other, and the overall structure of the story. These labels cannot be identified reliably from single keywords, which is why even more advanced models that use context sometimes struggle, leading to lower F1-scores.

3.3 Comparison of the Models

The logistic regression model has the advantage of being very simple both in terms of implementation and computation. It trains and predicts very quickly, providing an efficient and accessible baseline. However, its performance in terms of precision and F1-score is generally lower, as it cannot fully capture contextual information.

The transformer-based model, while more complex in design, remains computationally efficient thanks to its single-layer structure. It does not require much more time than logistic regression, yet it achieves substantially higher accuracy (93.98% on the test set) and stronger F1-scores for several labels, such as *BadEnding*, *Dialogue*, and *Twist*. This shows that the model is able to leverage contextual information effectively without sacrificing speed.

Among the models evaluated, distilBERT achieved the highest accuracy, outperforming the other two approaches. Nevertheless, its main limitation lies in the training time: fine-tuning distilBERT required approximately 200 minutes, whereas the other models completed the same process in about 2 and 4 minutes, respectively. This substantial difference can be attributed to the much larger number of parameters in distilBERT compared to the smaller models.
Overall, this makes distilBERT the best-performing model in terms of predictive accuracy and balanced performance across all labels.

3.4 Impact of Model Depth

We also experimented with increasing the number of layers in the second model, but this did not lead to significant improvements in performance, while noticeably increasing computation time. This suggests that the current single-layer encoder provides an optimal trade-off: it is robust and effective, delivering strong results without introducing unnecessary complexity or computational cost.