Improving Writing Assistance at JetBrains AI

Internship Task

Michal Svec

Introduction

This report compares selected methods for classifying text as formal or informal. For all experiments, we selected three models and a dataset containing French and English text samples. All code required to reproduce our experiments is located in the GitHub repository.

Dataset

As a dataset we chose FAME-MT. It is a dataset of translations between 15 source and 8 target languages labeled as formal or informal. Initially, we wanted to do the experiments also for Slovak language because it is our native language, and we wanted to know whether it can be a significant hurdle for certain models. Eventually, we used dataset with French text samples because there are pretrained models that can also work with French language.

The dataset contains 50000 text samples in French and English language. For this task we decided to create a smaller dataset of 5000 records where each class is balanced. Then, we used split 80% (4000 records) for training, and the remaining 20% (1000 records) is used for testing. It is important to note that the split is stratified.

We did not use a validation dataset for tuning hyperparameters, as it was only possible to tune a single model. From our experience, it usually brings only minor improvements, and we did not want to spend time in this area.

Models

XGBClassifier

A decision tree based model that uses eXtreme Gradient Boosting (XGBoost) algorithm, which is an optimized implementation of gradient boosting. The reason we chose this method is that we wanted to try a more traditional ML approach and create a contrast with more modern methods. It is known that tree based methods tend to perform well on tabular data. We used very rudimentary approach to convert the text into tabular data. The Snowball stemmer and stop words from the *NLTK* library were employed, the bag-of-words representation was obtained using the *scikit-learn* library.

XLM-Roberta-based classifier

This model was chosen because it was pretrained on a multilingual formality classification dataset and also supports the French language. We took this model from HuggingFace and leveraged CUDA on T4 GPU in Google Colab for faster inference.

Gemini 2.5 Pro Experimental

We chose this model because based on LMArena it should be the best model at this time. Initially, our plan was to use the API but we quickly hit rate limits so we decided to take another approach. It is quite unusual. Models from Gemini family are known for their large context window. We just saved the test split of 1000 records into file and pasted that text directly into the model through user interface of Google AI Studio. The model probably had trouble distinguishing between new samples and continuations. Therefore, we decided to add *SEP* (separator) to each text sample. After this, the model correctly outputted 1000 predictions.

This was the used prompt:

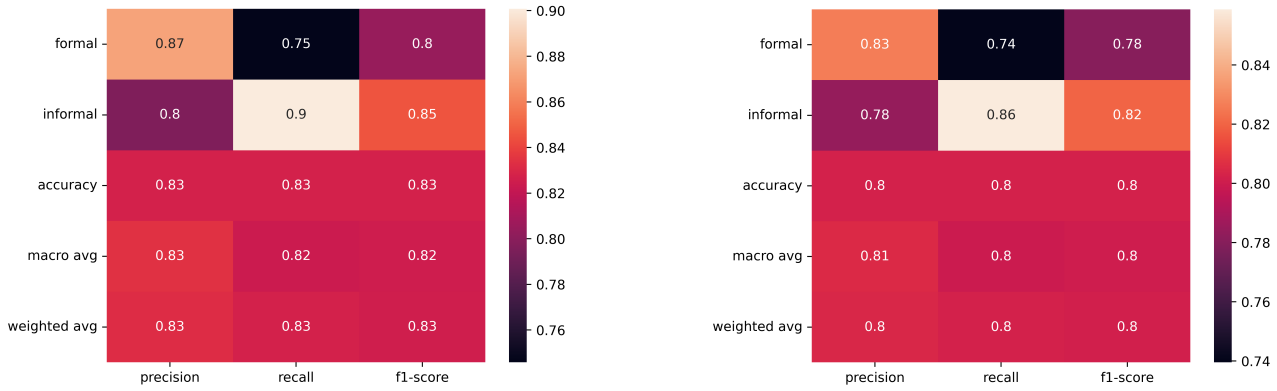
[1000 lines of data]

YOUR TASK:
*Each text sample is separated with *SEP*. Your task is to classify this text into two categories. Output 0 if formal, 1 if informal. There is 1000 text samples. Output the result as a python list with 1000 elements of 1 or 0.*

Results

XGBClassifier

Strikingly, the more standard ML approach achieved the best accuracy. The model performs slightly better in English language than in French, but the difference is minimal. It even surpasses XLM-Roberta, which was pretrained on a formality dataset. Most of the recall and precision scores are around the 0.80 level.

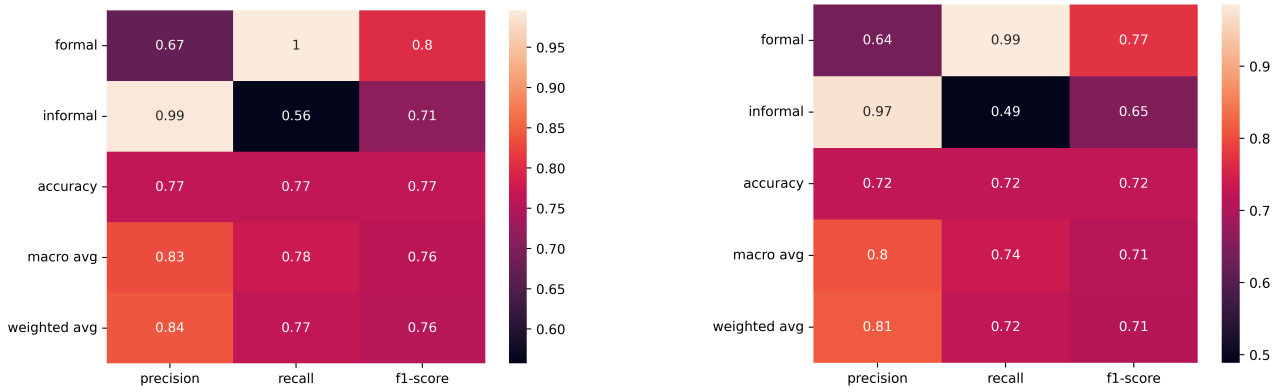


(a) English language (b) French language

Figure 1: XGBClassifier classification report, default parameters, BoW 1000 features

XLM-Roberta-based classifier

This model reached 77% accuracy for English language and 72% accuracy for French language. An interesting thing is that both models have very high recall therefore they are perfect at identifying formal sentences but they struggle with informal ones.



(a) English language (b) French language

Figure 2: XLM-Roberta classification report, default configuration from HuggingFace

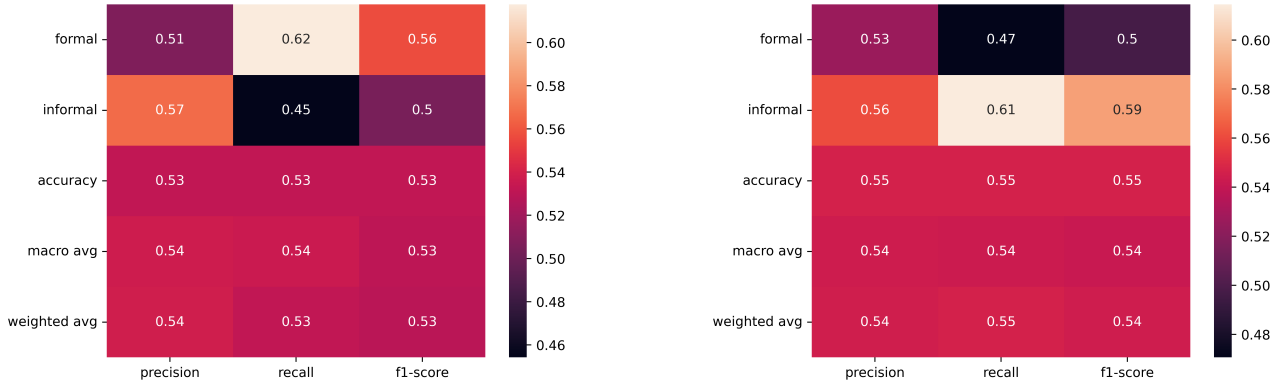
Gemini 2.5 Pro Experimental

We were wondering whether to include this unsuccessful attempt into this report. The models reached only about 53% accuracy. That is just 3% better than a simple baseline model that would just predict a single class.

We tried the prediction multiple times, but the accuracy remained roughly the same. This result was very surprising. When we looked at the model's output, it seemed promising, but the opposite was true.

We think that when classifying each sample separately over the API, the model could achieve better performance. We suspect the failure of this approach could be due to the output process. The LLM was supposed to output a single vector of 1,000 values (0 and 1), but there is a chance that the model somehow messed up that vector and therefore entire classification was incorrect.

The moral of the story is that we cannot rely on the model in this "bulk classification" approach because issues can arise when the model is producing the output. Despite the bad results, we have at least learned something.



(a) English language

(b) French language

Figure 3: Gemini 2.5 Pro Experimental classification report, used in Google AI Studio

Conclusion

We presented three models and two of them were usable. We argue that the poor performance of the Gemini model was primarily caused by our approach of using "bulk classification" instead of the API. In hindsight, we could have used APIs from Groq, as they offer a free tier. From the above experiments, it is clear that the winner is XGBClassifier. Its performance can be probably further improved with use of TF-IDF, embeddings and hyperparameter tuning. Furthermore, this model is lightweight to run in opposed to other models we tried.

We are aware of the fact that all work done here could be done better (larger dataset, explore more methods, hyperparameter tuning, more thorough evaluation...) but we also wanted to complete it within reasonable timeframe...