ANLP Competition Report - Language Classifier

Abstract

In this project, we have developed a text language classifier capable of automatically identifying the language of a given text. Our approach is based on the use of pre-trained Transformer models, with different performance enhancement techniques such as cross-validation, data augmentation and early stopping. We compared several model configurations to assess the impact of different features on final performance.

1 Introduction

Automatic language classification is a fundamental task in automatic natural language processing (ANLP). With the linguistic diversity present in modern datasets, it is crucial to develop robust systems capable of reliably identifying the language of a text. In this project, we developed a classification system based on pre-trained Transformer models. Our aim was to explore the impact of various preprocessing, data augmentation and hyperparameter tuning techniques to improve performance.

2 Exploratory Data Analysis

2.1 Overview

The data used in this project comes from the file `train_submission.csv`. We have a very large dataset, containing 190599 instances for 390 classes. Class labels correspond to the ISO 639 standard, which uses codes of 2, 3 or 4 letters to represent language names.

Les principales caractéristiques du dataset sont:

2.2 Classes Distribution

The dataset appears to follow a uniform distribution: 86% of classes contain exactly 500 instances. On the other hand, there is a strong imbalance for 15 classes with less than 10 instances, and 18 classes with more than 1000 instances. In particular, the Tadgik label “tgk” is over-represented with 1,500 instances. The presence of very rare classes is likely to make their classification difficult, in the absence of sufficient examples to generalize correctly.

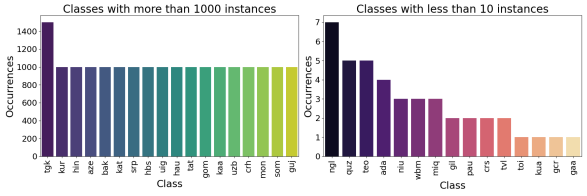

Figure 1: Sample of unbalanced classes

2.3 Length of the texts

Around 55 % of the sentences in the dataset contain less than 1000 words, while the remaining 45 % span a wide range of lengths, from 1000 to 78,939 words. However, these long sentences are infrequent, with the number of occurrences falling sharply as length increases. Figure 2 shows the distribution of text lengths.

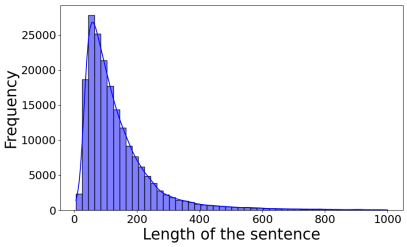

Figure 2: Distribution of number of samples by language

3 Solution

3.1 Data pre-processing

Several cleaning operations were applied:

- **URL deletion** (optional depending on configuration).
- Removal of **frequent occurrences** to reduce noise.

- **Stratification** when splitting data to ensure a balanced distribution of classes.

3.2 Data Augmentation

To address the class imbalance, we first generated additional examples for labels with fewer than 20 occurrences in the dataset by randomly shuffling words within existing examples until reaching 20 instances. Then, we tried to go further by applying data augmentation techniques using the `nlpaug` library (typo insertion - 30%, word deletion - 30%, and word swapping - 40%). However, this approach did not particularly yield better results. We also thought of splitting the samples of classes with few occurrences per sentence to increase the samples, but the grammatical specificities of each language make this process complex (no space between words, different punctuation...).

3.3 Train Test Val Split

In order to evaluate and compare our models and functionnalities, we made a train test val split with a `seed = 42`. We chose 80% of data for train, 10% for validation and 10% for test. Additionnaly we used the parameter `stratify` to keep the same proportion of label in every dataset. It avoids having one class only in val set and not in train especially for label with little occurences.

3.4 Training

We used as classification model: `intfloat/multilingual-e5-large-instruct`. Training was carried out on a Tesla V100 (32Go) graphics card with the following parameters:

Hyperparameter	Value
Learning Rate	3×10^{-5}
Batch Size	128
Epochs	5
Optimizer	Adam
Early Stopping	2 epochs

Table 1: Hyperparameters used for training

We also performed a 3-fold StratifiedKFold cross-validation to ensure that our model generalises well and to finetune these hyperparameters. Concerning training time, one epoch took around 10 to 15 minutes. Two methods were used to log the results:

- the automatic logging of `mlflow` (seen in TP)
- a custom logging that provides an overview of hyperaparameters and metrics to compare models

Best model

The best model used for submission was trained on the entire dataset (train + val + test) with the same hyperparameters as in Table 1, except for the number of epochs, which was increased by two to a total of 7 to account for the additional data.

4 Results and analysis

4.1 Models comparison

We have tested different architectures for a basic configuration, with the following results (limited number of training epochs) :

Modèle	F1-Score on test set
`intfloat/multilingual-e5-large-instruct`	**0.89**
`distilbert-base-multilingual-cased`	0.85
`bert-base-multilingual-cased`	0.82
`xlm-roberta-base`	0.84

Table 2: Comparaison des modèles sur la configuration de base.

As we can see, the **multilingual-e5-large-instruct** model is better, hence the choice of this model for hyperparameter tuning and final training.

4.2 Feature analysis

We tested the impact of various features on the final performance of the selected model:

Functionality	F1-Score on test set
Occurrence Removed, No Weight Decay	**0.8989**
No Occurrence Removed, No Weight Decay	0.8912
Occurrence Removed, With Weight Decay = 0.01	0.8977

Table 3: F1-Scores for Different Functionality Choices

Therefore, we obtain best results when removing occurences of label where number of instances is greater than 500 and reduce the number to 500 (Occurrence Removed). Also having `weight_decay` equal to 0 gives better results.

Finally, we note that the model's prediction errors mainly concern classes present less than 10 times in the dataset, although we applied a data augmentation to reach 10 occurrences per class, recall and precision remain at 0 for these categories.

5 Conclusion

Our best model "multilingual-e5-large-instruct" gave us an accuracy of 0.89635, putting us in 2nd

place on the leaderboard of the Kaggle Competition.

6 References

- Jacob Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2019.
- Yinhan Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach", 2019.
- Liang Wang et al., "Multilingual E5 Text Embeddings: A Technical Report"