Languages classification

Coz Erell and **Lançon Camille** and **Mangin Elian** and **Leene Arthur** and **Khlaut Hugo**

Abstract

In this paper, we propose a machine learning-based approach to language classification using a dataset of sentences in multiple languages. We explore different feature-augmentation techniques such as word embeddings and alphabet recognition to help us on this task. We performed a multimonial naive bayes approach as well as approaches using pre-trained model with FastText, Roberta and a multi-model approach.

1 Introduction

Language identification is a fundamental task in Natural Language Processing (NLP) that involves determining the language of a given text. Accurate language classification has downstream applications, including machine translation, sentiment analysis, and multilingual information retrieval. We have at our disposal a dataset of **38,884** entries with the following structure.

Columns	Non-Null Count	Unique val count
Id	38854	80
Usage	38854	1
Text	38854	38852
Label	38854	389

Table 1: Dataset structure

There are a total of **389** classes. We observe in 1 that some classes are more frequent than others. Some languages have less than 3 instances in the whole dataset, predicting correctly those languages is a task next to impossible. Most classes have a number of instances between 80 and 100 as depicted in 2.

2 Solution

2.1 Alphabets

We attached a corresponding alphabet to each instance. Of the **38,854** instances, we were able to detect the alphabet of **37,628** instances. We use the following alphabets : Latin Cyrillic, Arabic, Hebrew, Chinese, Hiragana, Katakana, Korean, Greek, Gujarati, Devanagari, Thai and Georgian. Some examples of the dataset contain different alphabets (see the Example Appendix).

2.2 Embedding

We used a Hugging Face SBERT model to embed our data. We chose the *distiluse base multilingual-cased-v2* model to obtain embeddings close to each other for the same language and with a maximum distance for two different languages. We obtained vectors of size 512. We attempted to group languages within clusters of 2D dimensions thanks to a PCA or T-SNE approach, but it did not provide satisfying results as depicted in 3. Two dimensions might not be sufficient to separate the data.

2.3 Models

After embedding the different sentences, we tested different models to identify the language of the texts:

- Multinomial Naïve Bayes Approach
- FastText model
- Roberta with finetuning
- Multi-models approach

In view of the results obtained on the different models, we finally decided to keep the Roberta model for the test phase.
The next section details the results obtained with the different models.

3 Results and Analysis

3.1 Tiny Alphabets

We observe that some alphabets only include one language. Therefore, if one of these alphabets is

detected, it can be associated with a language. We obtain the following associations between alphabets and languages.

- Thai -> 'tha'
- Hiragana -> 'jpn'
- Gujarati -> 'guj'
- Korean -> 'kor'
- Katana -> 'jpn'

3.2 Small Alphabets

Some alphabets contain less than 5 languages. That is the case for the Hebrew, Greek and Georgian alphabets. A random forest model in the embeddings is sufficient to distinguish languages within those alphabets. After splitting the training data in 80% training and 20% testing, we obtain the confusion matrix in 4, 5, 6. The results encourage using random forest models for those alphabets.

3.3 Bigger Alphabets

3.4 Multinomial Naïve Bayes Algorithm

We first tried a simple approach with a multinomial naïve Bayes algorithm.
The tokenization was done by splitting on whitespace and removing punctuation, and the embeddings calculated by counting the tokens: each sentence is then represented as a vector of token counts. The length of the vector is equal to the size of the vocabulary.
We trained a Bayes algorithm based on Bayes' theorem:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

with C the target class/label and X a token.

For the training phase, we compute the prior probability $P(C)$ for each class C (i.e the frequency of each class in the training data). Then, compute the likelihood $P(X|C)$ for each token given each class (i.e the frequency of each token in sentences of class C normalized by the total number of tokens in class C).
For the prediction phase, for a given sentence, we compute the posterior probability for each class using Bayes' theorem and we assign the sentence to the class with the highest posterior probability.

Multinomial Naive Bayes Algorithm: Using an 80/20 train-test-split, the model achieved an accuracy of 0.74 and a F1-score of 0.73.

3.5 Training a FastText model

We used the FastText library to train a model capable of classifying languages based on our dataset. First, we split the dataset according to the different alphabets to train specialized models and using an 80/20 train-test split for each alphabet. After some hyperparameter optimization, we achieved good results on alphabets such as Cyrillic and "Inconnu" (a category grouping all unrecognized alphabets), with test precisions of 0.84 and 0.90, respectively. However, performance was lower on the Latin and Arabic alphabets, with test precisions of 0.76 and 0.59, respectively. We also observed that the models tended to overfit.

One key advantage of FastText is that it allows for fast training and inference on any device, which motivated us to experiment with it. However, given the performance limitations, specifically for the Latin alphabet (which has the largest number of instances), we decided to explore a more effective approach.

3.6 Fine tuning Roberta for language classification

This approach leverages the **XLM-RoBERTa** model, a multilingual variant of the RoBERTa transformer, to perform language classification on our dataset. XLM-RoBERTa is pre-trained on a vast corpus of multilingual text using a self-supervised masked language modeling (MLM) objective, making it well suited for cross-lingual tasks. We fine-tune the *xlm-roberta-base* model on our dataset, optimizing classification accuracy while handling significant class imbalance using weighted loss functions. The input text is tokenized using **subword tokenization**, ensuring robust representation across different languages and scripts. During training, we implement **early stopping** and evaluate performance using the **weighted F1-score** to select the best model. Using an **80/20 train-test split**, our fine-tuned model achieved a **weighted F1-score of 0.83**, demonstrating its effectiveness in handling diverse language classification challenges.

3.7 Multi-models approach

We combined the above approaches for tiny and small alphabets with Roberta's results for larger

alphabets only.

Alphabet	Model
TINY	Direct association with one language
SMALL	Random Forest
LARGE	Roberta

Table 2: Dataset structure

However, the results have not improved with this approach and remain at 83% for F1-score like Roberta.

3.8 Appendices

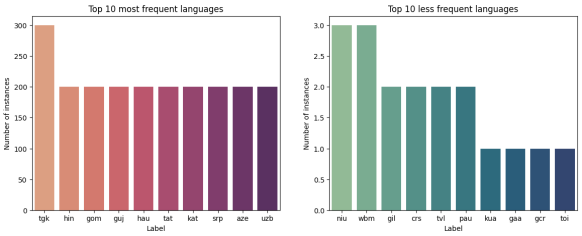

Figure 1: Class frequencies

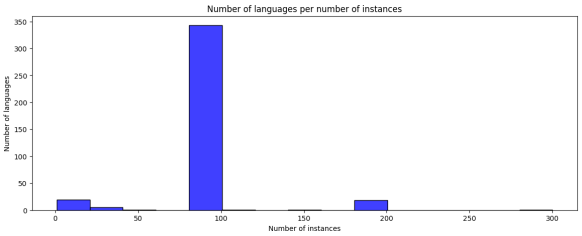

Figure 2: Number of languages per number of instances

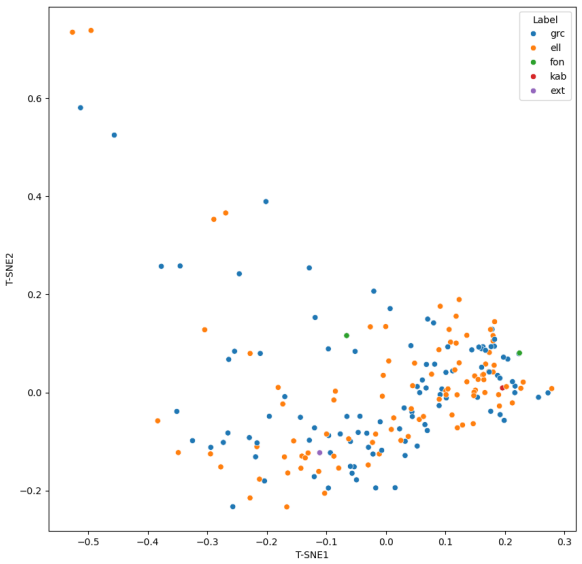

Figure 3: T-SNE approach for Greek languages

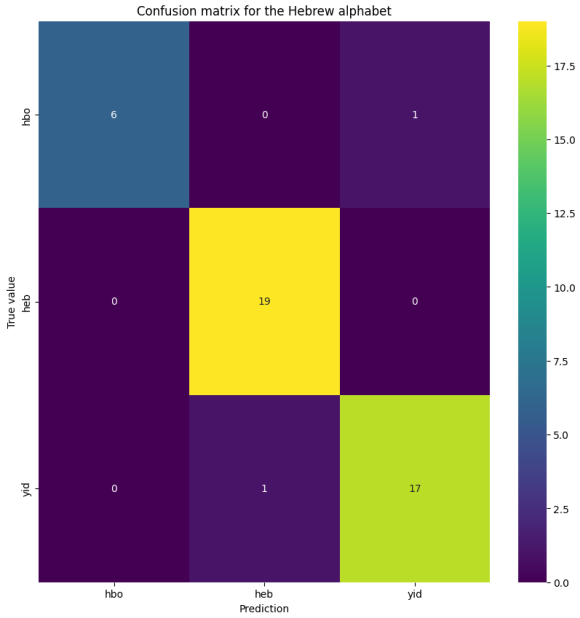

Figure 4: Confusion Matrix Hebrew Alphabet

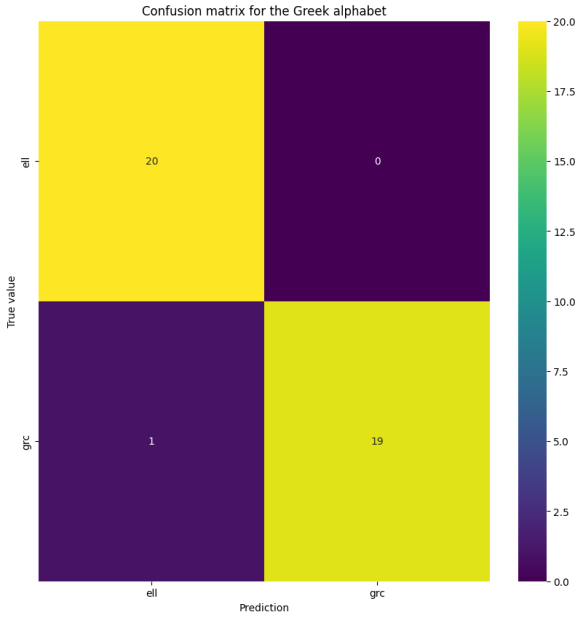

Figure 5: Confusion Matrix Greek Alphabet

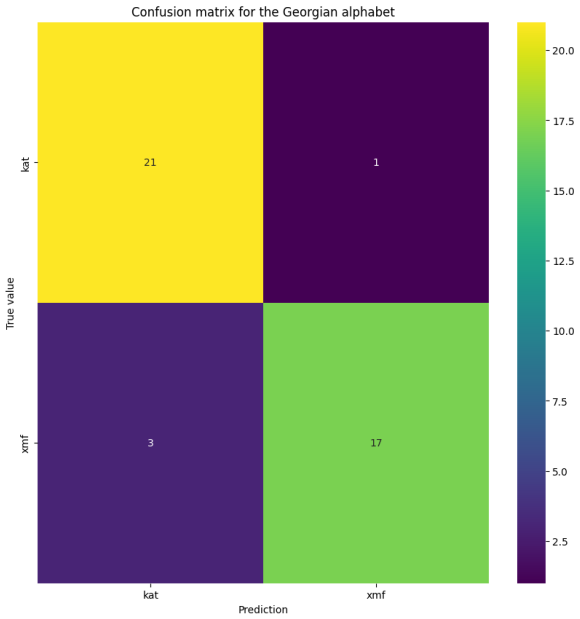

Figure 6: Confusion Matrix Georgian Alphabet