A Simple Language Classifier

Marius Nadalin

Abstract

This paper presents a machine learning approach to automatic language classification of text. This work explores different preprocessing steps such as alphabet extraction, and optimizes a classification model to identify the language of a given text. The final implementation uses alphabet-specialized TF-IDF vectorization and logistic regression, optimized using grid search. The results demonstrate the effectiveness of the approach in the classification of multilingual texts.

1 Introduction

Language classification is a fundamental task in Natural Language Processing (NLP) with applications in multilingual information retrieval, machine translation, and automated content filtering. In fact, state-of-the-art translation models make use of automatic language detection of a given text before applying a specific bilingual translation sub-model. The challenge lies in distinguishing languages based on textual features, especially for closely related languages. Thus, this work aims at exploring and tuning the implementation of ML-based language classification models.

2 Approach and Methodology

2.1 Data Exploration and Preprocessing

In the context of a Kaggle challenge, a labeled training set containing texts in multiple languages is provided. An initial exploration reveals that this training set contains an unbalanced representation of 390 languages.

Consequently, a first preparation step was to balance the data to maintain an equal number of texts per language by filtering out classes (languages) with too few samples and limiting the number of texts for over-represented classes. This ensures that the model learns equally from all languages included, preventing bias toward any particular one.

Then, one can notice that detecting the script (writing system) of a given text provides a valuable preprocessing step. We implement a script detection method that identifies the dominant alphabet of a text using Unicode character ranges for major scripts, such as Latin, Cyrillic, Arabic, Devanagari, and Chinese. By determining the most frequent script in a text, we can refine the classification pipeline by narrowing down the possible language candidates, thereby improving efficiency and accuracy.

Additionally, texts with ambiguous or rare scripts (where no single script dominates) can be handled separately. These texts are classified as belonging to a "rare alphabet" when their most dominant script accounts for less than a predefined threshold of total characters. In our implementation, empirical tests led us to set this threshold at **40%**, meaning that if no single script contributes to at least 40% of the characters in a text, it is flagged as a "rare alphabet". This approach improves the robustness of our classification pipeline, ensuring that ambiguous or uncommon texts are treated separately to reduce misclassification errors. Table 1 shows the distribution of texts induced by this script classification.

Script (alphabet)	Percentage
Latin	72.37
Cyrillic	8.53
Other	8.12
Arabic	6.49
Devanagari	3.14
Chinese	1.36

Table 1: Script distribution in the training set with a rare language ('Other') threshold of 0.4.

2.2 Model

Our language classification pipeline consists of multiple stages:

1. **Script Detection:** The primary script of each text is identified using Unicode character ranges. This step helps refine classification by narrowing down possible languages and assigning a script-specialized classifier.

2. **MultiScript Model:** A separate classifier is trained for each major script category (e.g.,

Latin, Cyrillic, Arabic). The detected script determines which classifier is used for a given text.

3. **Feature Extraction:** A script-specialized TF-IDF vectorization is applied to convert textual data into numerical form, highlighting important words while reducing noise.

4. **Feature rescaling:** A standard scaler is used to normalize the feature values. It is useful because logistic regression is sensitive to the scale of the input features, and scaling helps improve model performance by stabilizing convergence and enhancing accuracy.

5. **Final Classification:** A logistic regression classifier is trained using the TF-IDF features.

2.3 Model Training and Optimization

To enhance performance, we apply grid search with cross-validation to optimize hyperparameters. This ensures the best-performing model is selected for final evaluation.

Firstly, we refined the TF-IDF vectorization. This method converts textual data into numerical vectors by considering the relative importance of words within each document. Words frequently occurring across all documents are downweighted, while those unique to specific languages receive higher significance. Important parameters are the analyzer mode (whether to use words or character n-grams), the range of the extracted n-grams, and the maximum number of features to keep in the vectorizer. Grid-search experiments led us to use a **character-level analyzer**, n-grams ranging from **unigrams** to **trigrams**, and a maximum of **15,000 features** depending on the associated script.

Secondly, we tuned the logistic regression classifiers. Empirical experiments led us to use a regularization parameter $\mathbf{C = 10.0}$.

Thirdly, a grid search optimization process regarding the script detection determined that a **rare script threshold of 0.4** is the most effective hyperparameter setting for our needs.

Finally, we conducted empirical experiments to find the best script combinations to keep in the multilingual classifier. This is an important part since each script will be associated with one specialized TF-IDF vectorizer and logistic regression classifier. The results presented in Figure 1 led us to separate the **Latin**, **Cyrillic**, **Devanagari**, and **Chinese** scripts (alphabets) apart and to group all other scripts together.

3 Results and Analysis

The trained model is evaluated on a held-out test set, obtained using a stratified train-test split strategy (80-20) on the provided training set. We use accuracy as the primary metric to measure classification performance. The multilingual classification model presented in the previous section achieves a **mean accuracy of 0.83**, using a five-fold cross-validation method.

Since we limited the architecture to relatively small components (less than 15,000 features per TF-IDF vectorizer), this lightweight model has a size of **56.34 MB**.

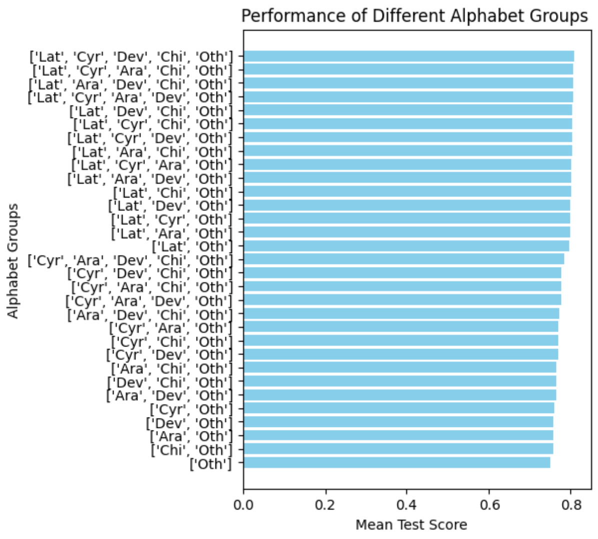

Figure 1: Grid-Search and Cross-Validation results on the script-based separation for the multilingual classification model. 'Lat' refers to Latin, 'Cyr' to Cyrillic, 'Dev' to Devanagari, 'Chi' to Chinese, 'Ara' to Arabic, and 'Oth' to Other (i.e., the rare alphabets).

4 Conclusion

This study demonstrates that logistic regression combined with TF-IDF can effectively classify languages based on textual input, with promising accuracy scores across multiple languages.

Future work may explore deep learning approaches and larger datasets to improve classification accuracy further.