Practical implement language detection for Laos, Vietnamese and other languages

Ta Nguyen Thanh
Institute for Artificial Intelligence
UET-VNU
23020437@vnu.edu.vn

October 2, 2025

1 Introduction

Language Identification is a foundational task in Natural Language Processing (NLP), involving the automatic classification of a given text segment into its corresponding source language. This process is essential for many downstream tasks, including machine translation, information retrieval, and targeted content filtering. The core challenge in LID is developing models that can robustly distinguish between languages based on intrinsic characteristics of their written form.

The differentiation of Southeast Asian languages presents unique and compelling challenges for language identification. Vietnamese, for instance, employs the Latin script with an extensive system of diacritics and tone marks, while Lao is written in an entirely distinct Abugida script that requires specialized normalization and encoding strategies. Beyond these two cases, other regional languages introduce their own structural and orthographic complexities, ranging from diverse scripts to varying degrees of diacritic usage. By considering multiple languages rather than focusing solely on a single pair, this work aims to develop more generalizable techniques that capture the broader variability of writing systems, ensuring that the language identification model remains robust and adaptable across different linguistic contexts.

Following the established methods in the field, this problem is primarily approached as a text categorization task, often leveraging statistical models based on character n-grams. This methodology, which analyzes sequences of characters to create a probabilistic linguistic fingerprint, is a widely adopted technique for language identification (Hidalgo, 2013). The objective of this report is to

evaluate these classication methods for a variety of distinct languages.

2 Data

For this assignment, Lao data is collected from this dataset on Kaggle. All remaining languages are crawled from The Leipzig Corpora Collection, using material in various news website in 2019. Those languages are Vietnamese, English, French, German, Arabic and Russian. For each specific language, there are 300.000 sentences, each is on a separate line of text. The data is first loaded into a **Pandas DataFrame**, then it is preprocessed carefully before written back to a **train.txt** file and a **test.txt** file with an 80/20 ratio. In order to preprocess the Lao language, I've used several methods such as collapsing multiple whitespace into a single space, removing control characters such as U+200E, pruning all special characters like emojis, invisible spaces, …For the other languages including Vietnamese, I applied the same techniques to make sure all training and test data is normalized, avoiding skewness and imbalance. Before writing the data back to text files, I also added labels in the training file to facilitate training with **scikit-learn** afterwards.

3 Methodology

3.1 Feature Extraction

For this study, language identification is treated as a text categorization task. We employ character-level *n-grams* as features, since short character sequences effectively capture orthographic and morphological patterns that differ across languages. Each document is represented using Term Frequency–Inverse Document Frequency (TF-IDF) weighting, implemented with the scikit-learn version of the Vectorizer, where I've used character n-grams in the range 1-3, comprising of unigrams, bigrams and trigrams. The maximum features for each sentence are clipped from 278.000 to 50.000 to avoid RAM overload, as well as to reduce training time and increase overall accuracy. The TF-IDF value of term t in document d is defined as:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \cdot \log \frac{N}{\text{df}(t)},$$

where $\text{tf}(t, d)$ is the frequency of t in d, N is the total number of documents, and $\text{df}(t)$ is the number of documents containing t. The resulting TF-IDF matrix is a sparse matrix with the shape of (N, F) where N is the number of sentences that was fed, and F is the number of features extracted in the shape of a vector. In this case, N = 1.680.000 and F = 50.000.

3.2 Models

We use two supervised classifiers:

- **Naive Bayes (Multinomial)**: This model assumes conditional independence of features and is well-suited for categorical count-based features such as character n-grams. The probability of class c given document d is:

$$P(c \mid d) \propto P(c) \prod_{i=1}^{n} P(f_i \mid c),$$

where f_i are the observed features.

- **Logistic Regression**: A linear model that estimates class probabilities via the sigmoid function. For input vector $\mathbf{x}$ and weights $\mathbf{w}$, the probability of class c is:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \quad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Logistic regression works well with high-dimensional sparse features such as TF-IDF.

3.3 Implementation Details

The dataset is randomly shuffled and divided into training and testing sets with an 80/20 ratio to ensure robust evaluation. Therefore, the train set contains 240.000 sentences and the test set contains 60.000 sentences. After having two files **train.txt** and **test.txt**, they are read into **Pandas DataFrames** again, and then the training DataFrame is shuffled with a determined seed to ensure the model doesn't bias any languages, and its results can be replicated. In this assignment, I've used **scikit-learn** library for almost all of the training, predicting and evaluating functions, including both a classification report and a confusion matrix.

4 Experiments

All the code can be found in this GitHub repository. I've trained the model with two different classifiers: The Naive Bayes classifier and the Logistic Regression classifier, and the results are outstanding. For Naive Bayes, the model achieved a remarkable accuracy of **99.91%** within only **2 minutes** of training time. For Logistic Regression, the model achieved a slightly better accuracy of **99.95%**, with a training time of approximately **5 minutes**. Using **scikit-learn's classification_report** function, we can deduce that in both cases, the precision, recall and F1-score metrics are **1.00** for all languages. The confusion matrices of two models can be seen in Figure 1, which are created using **scikit-learn's confusion_matrix** function. After evaluating, I fed the models with some example sentences like in Table 1 and collected the predicted label for each one.

```
Confusion Matrix:
[[59999     0     0     1     0     0     0]
 [    0 59879   114     7     0     0     0]
 [    0     7 59978    14     0     1     0]
 [    0    24   131 59845     0     0     0]
 [    0     5    32     1 59960     0     2]
 [    0     0     1     0     0 59999     0]
 [    0     2    23     0     0     0 59975]]
```

(a) Naive Bayes model's confusion matrix

```
Confusion Matrix:
[[60000     0     0     0     0     0     0]
 [    0 59942    52     6     0     0     0]
 [    0     6 59979    11     2     1     1]
 [    0    21    53 59923     3     0     0]
 [    0     2    12     2 59981     0     3]
 [    0     0     1     0     0 59999     0]
 [    0     2    14     1     0     0 59983]]
```

(b) Logistic Regression model's confusion matrix

Figure 1: Confusion Matrices of two models

Test Sentence	Predicted Language
Bonjour, comment ça va ?	fr
Xin chào, bạn khỏe không?	vi
Hello, how are you?	en
Привет, как дела?	ru
Guten Tag, wie geht es Ihnen?	de
مرحبا، كيف حالك؟	ar
ສະບາຍດີ, ເຈົ້າສະບາຍດີບໍ?	lo

Table 1: Examples of test sentences and their predicted languages.

5 Conclusion

In overall, with a very high accuracy, we can see that the model has an excellent performance for a training set of 1.680.000 sentences in total. There is always potential for further improvements in the future, such as collecting more data, up to 1 million sentences or more per language with more unique languages, using a larger n-gram window of 4 or 5, or using deep learning libraries to leverage training process with GPU utilization to help reduce training time and data bottleneck. In the training process, we can also use Principle Component Analysis (PCA) for dimensional reduction, as well as an auto scaler to normalize or rescale the features so that they are on a comparable scale. We believe that with several functional improvements, careful optimizations and fine-tuning, the model can achieve better results for much larger dataasets.

References

[1] J. M. Gomez Hidalgo, *Language Identification as Text Categorization*, Blog post, 2013. https://jmgomezhidalgo.blogspot.com/2013/05/language-identification-as-text.html

[2] Leipzig Corpora Collection, *Leipzig Corpora Collection*, 2025. https://wortschatz.uni-leipzig.de/en/download

[3] chickooo, *NLTK vs spaCy | Which NLP library to use?*, Kaggle Kernel, 2023. https://www.kaggle.com/code/chickooo/nltk-vs-spacy-which-nlp-library-to-use#Conclusion

[4] CodezUp, *Building a Text Classification Model from Scratch with Python*, CodezUp, 2025. https://codezup.com/building-text-classification-model-python/