Master in Data Science

D1: Language Detection

Mining Unstructured Data

Year
2023/24

Date

Team members
Alícia Chimeno Sarabia and Gabriel Vayá Abad

Introduction

According to Linguists, there are officially 7.139 known languages worldwide, some of them utilizing diverse writing systems. These writing systems are based on the representation of language sounds through symbols, characters, or letters, and each language employs its unique method of transcription.

Recognizing a language from a text can be quite challenging for a human not only because of the number of different alphabets but also because of the vast array of dialects, slang, and regional variations within languages themselves. Consequently, language identification often requires a combination of linguistic expertise, contextual clues...

In this project, we're immersing ourselves in the realm of natural language processing (NLP), with a specific focus on **language detection**. Our objective isn't just to build a robust language detection program but also to methodically experiment with various parameters to refine our model. Through this process, we aim to gain a deeper understanding of the outcomes and critically analyze the underlying rationale.

1 Preprocess

1.1 Dataset exploration

Before applying any machine learning algorithm and optimizing it, our initial focus was on enhancing the quality of the data through some preprocessing steps. Initially, we explore the dataset to identify areas where data quality could be improved, such as refining sentences or making slight adjustments to the data to enhance classifier performance. We have noted the following points during our exploration that we should consider on the preprocessing:

- The dataset is made of 22.000 sentences from 22 different languages. This languages come from multiple root languages, and have different alphabets and structures.
- We have a data quality problem. The dataset has sentences that are not classified well, therefore this would affect on the results. We found several examples, one would be

  > "military justice as in the uniform code of military justice most nations have a separate code of law which regulates both certain activities allowed only in war as well as provides a code of law applicable only to a soldier in war or in uniform during peacetime" - Pushto

- There are some sentences that have mixed languages, mainly sentences in one language that have brackets or quotation marks that translate the sentence into another language.
- There are languages that use no spaces like Chinese or Japanese, so we suspect that different preprocessing methods will be required by these kind of languages.

2 Code

Code attached in a Python file. (readme)

3 Experiments and results

We consider the **F-score**, the **Confusion Matrix** and the **Coverage** as indicators for the optimal model for predicting language from text inputs. Also we will use the **Recall** of a specific language to detail more our analysis[1] and also the **False Positive Rate**[2].

Our objective is to train the most effective model by experimenting with various parameters such as:

- **Levels**. This refers to the granularity at which the text data is analyzed. It can be either at the **character** level or at the **word** level.
- **Vocabulary size**. Number of tokens in the vocabulary: 1000.
- **Machine Learning Algorithms**: Naive Bayes algorithm, Support Vector Machine, Decision Trees.
- **Preprocess methods**: Remove expressions between parenthesis (and quotation marks, dashes and "guillemets"), remove punctuation, sentence splitting.

Each experiment will be thoroughly examined to discern why certain approaches perform better or worse than others. We will use the following notation to compare results, f will be our model classifier and the parameters that we will vary are f(Level, Vocabulary Size, ML Algorithm, Preprocess Steps)

3.1 Experiment 1: different levels

In our first experiment, we aim to compare the vocabulary coverage and performance depending on the analyzer granularity (character or word) and explain why show different error patterns. For this preliminary investigation, we fix the following parameters: no preprocessing steps, limitation of the vocabulary to 1000 tokens, and utilization of the Naive Bayes machine learning algorithm, i.e $f(levels$,1000,Bayes, No Preprocess).

3.1.1 Character level

The model's character-level granularity reveals several key findings:

- Strong coverage. This suggests that the vocabulary effectively represents the dataset.

  ```
  Coverage:  0.9808517331929401
  ```

- At least 0.95 F-score. Good classification score overall.

  ```
  F1: 0.9552272727272727 (micro), 0.9578334706797794 (macro), 0.9574109803676453 (w)
  ```

- About the confusion matrix1. When we analyze the classifier for each language separately, we can observe that:

[1] $$recall(language) = \frac{\text{True language}}{\text{True language} + \text{False language}}$$

[2] $$FPR(\text{languageX, languageY}) = \frac{\text{Language X missclassified as Language Y}}{\text{True Language X}}$$

– Misslassification of Dutch into English. Although we have a high F-score, the recall from Dutch, Recall(Dutch)= 0.76. Most of the missclassification of Dutch are English, we can see by getting the following ratio,

$$\text{FPR(Dutch,English)} = \frac{\text{41 Dutch sentences missclassified into English}}{\text{230 Actual Dutch sentences}} = 0.18$$

.

– Sentences in English are 99,5% of the time classified good, recall(English) = 99.5. We notice that several languages are missclassified as English. This could be due to

  * The dataset quality is not high, as we said, there are sentences with the wrong language label, leading to the model may struggle to learn distinguishing features for those languages.

3.1.2 Word level

At the word level, we observe the following insights:

- Low coverage.

  ```
  Coverage:  0.25771498027437495
  ```

- 0.88 F-score.

  ```
  F1: 0.8920454545454546 (micro), 0.881225725511312 (macro), 0.8845797399266582 (w)
  ```

- Confusion matrix1. Upon closer examination of the confusion matrix, it becomes apparent that the classifier struggled with certain languages. Specifically, it misclassified many Chinese and Japanese sentences as Swedish. Proof of that, we can compute the recall of these specific languages,

$$recall(\text{Chinese}) = \frac{29}{29+10+1+1+1+1+157+1} = 0.14$$

$$recall(\text{Japanese}) = \frac{28}{10+1+4+2+1+144+1+3} = 0.17$$

  And

$$FPR(Chinese, Swedish) = \frac{157}{201} = 0.78$$

  Chinese and Japanese have in common that both languages use a combination of characters and symbols in their writing systems. They both belong to the East Asian language family, which includes other languages like Korean.

3.1.3 Conclusions

By applying the same inputs (size and machine learning algorithms), we can effectively compare both approaches by only varying the granularity of the data (character or word level). Clearly, the coverage significantly is high with character-level granularity (98%). This is because languages relying on alphabets often share characters between training and test sets, given that words are constructed from a limited set of letters. However, languages employing non-alphabetic characters, such as diagrams, may deviate from this trend, leading to a high but not flawless accuracy. The coverage of the word-level granularity is very low, and could be due to high variability between words due to:

- different alphabets, different languages.
- use of commas and other characters that may vary the vocabulary of the train.
- morphology (singular/plural, gender variations).

We will try to improve it using various preprocessing tasks.

3.2 Experiment 2: Vocabulary Size

In the previous section, we experimented with a vocabulary size of 1000. Now, we will explore a larger size for each granularity to assess potential variations in coverage and F-score. Furthermore, we aim to determine which languages are most affected by these changes.

3.2.1 Vocabulary size of 2000

Character level: f(Char, 2000, Bayes, No Preprocess)

- Higher coverage than when the vocabulary size is 1000.

```
Coverage:  0.9928643429019024
```

- 0.94, less F-score than when vocabulary size is 1000.

```
F1: 0.946590909090909 (micro), 0.9489551724489543 (macro), 0.9480278211312705 (w)
```

- Confusion Matrix3.
  - Worse classification of Dutch than when 1000 vocabulary size. The error is bigger, recall(Dutch)= 136/230= 0.59. And FPR(Dutch,English) =

Word level: f(Word, 2000, Bayes, No Preprocess)

- Higher coverage.

```
Coverage:  0.31814052278091903
```

- Same 0.89 F-score than 1000 words.

```
F1: 0.9034090909090909 (micro), 0.8952733711907622 (macro), 0.89824330473188 (w)
```

- Similar confusion Matrix4 as 1000 words.

3.3 Experiment 3: different machine learning algorithms

In our third experiment, we do not use any preprocessing steps, words as features, and constraining the vocabulary to 1000 words (fixed parameters). Our primary focus was on **varying the machine learning algorithms** while keeping these parameters constant.

3.3.1 NBA

Applying the Naive Bayes algorithm, the results are on section 3.1.2.

- Recall that the F-score minimum value is 0.88
- and the classifier misclassified many Chinese and Japanese sentences as Swedish.

3.3.2 SVM

Applying Support Vector Machine, we can asses that

- 0.92 F-score

```
F1: 0.9261363636363636 (micro), 0.9218315945824247 (macro), 0.9223382275476382 (w)
```

- The confusion matrix 5 shows that Chinese is being misclassified as Japanese.

3.3.3 Decision Tree

Applying decision trees modeling, we can see that

- The F-score

  ```
  F1: 0.9261363636363636 (micro), 0.9218315945824247 (macro), 0.9223382275476382 (w
  ```

- Taking a closer look into the confusion matrix5, we observe that: Chinese is misclassified, erroneously labeling Chinese sentences as Japanese. Computing the recall for Chinese, we can identify that the model struggles specifically with this language,

  $$recall(\text{Chinese}) = 0.164$$

  The other languages seem to work well enough.

3.4 Experiment 4: different preprocessing steps

In this section we have chosen the decision trees machine learning algorithm because of its high F-score and the program time running. We will asses the results of trying different preprocessing methods, taking as baseline case 1000 as vocabulary size, decision trees as machine learning algorithm and word level. We will also introduce a prior differentiation intended to filter those languages that do not use spaces, in order to use exclusive preprocessing techiques in them.

3.4.1 Sentence splitting, no differentiation

In this first experiments, we are making no differenciation between languages a priori, and we are splitting each sentence every 2 characters.

- The F-score

  ```
  F1: 0.845 (micro), 0.8451112737226627 (macro), 0.8457930799453445 (weighted)
  ```

- Looking at the confusion matrix6, we can see that the biggest problem is that we are misclassifying a lot of Korean sentences as Chinese.

3.4.2 Sentence splitting, remove punctuation, no differentiation

- The F-score

  ```
  F1: 0.8565909090909091 (micro), 0.8569894234897846 (macro), 0.8575562844583304 (weighted)
  ```

- In this case, the confusion matrix7 shows the same problem as before, even worsening it a little, and other minor problems arise in some indo-european languages.

3.4.3 Sentence splitting, remove punctuation, remove parenthesis, no differentiation

- The F-score

  ```
  F1: 0.8477272727272728 (micro), 0.8484109968084778 (macro), 0.8489657280391738 (weighted)
  ```

- The confusion matrix8 in this case shows that the classification of Korean as Chinese has been reduced slightly, however, now we can see that the algorithm labels a fair chunk of Portuguese sentences as Spanish.

At this point, it is clear that sentence splitting is not giving optimal results when applied to the whole dataset. Hence, in order to get better results, different preprocessing strategies have to be used in different languages, sentence splitting in particular. Now, to assess which of those need sentence splitting and which do not, we are considering that those sentences with less than 10 spaces, are those which have the potential to be benefited by this step.

3.4.4 Differentiation: Sentence splitting —— nothing

- The F-score

```
F1: 0.9318181818181818 (micro), 0.9336908697393613 (macro), 0.9339149569319248 (weighted)
```

- In the first case of this second group of experiments, we can see that the confusion matrix**??** shows an important improvement. We see some Korean sentences labeled as Chinese and also some Russian sentences labeled as Chinese as well. Nevertheless, the rest of languages are fairly accurately labeled.

3.4.5 Differentiation: Sentence splitting —— Remove punctuation

- The F-score

```
F1: 0.9327272727272727 (micro), 0.9346310380460835 (macro), 0.9348861111165518 (weighted)
```

- In this case, the confusion matrix looks very similar to the first case.

3.4.6 Differentiation: Sentence splitting —— Remove punctuation, remove parenthesis

- The F-score

```
F1: 0.9243181818181818 (micro), 0.9278707455338231 (macro), 0.9280785148034192 (weighted)
```

- We can see that the performance is very similar to the previous experiment, maybe with a slight decrease.

3.4.7 Differentiation: Sentence splitting, remove punctuation —— Remove punctuation, remove parenthesis

- The F-score

```
F1: 0.9252272727272727 (micro), 0.9289096145119949 (macro), 0.9290501253507633 (weighted)
```

- Similar as the previous experiment.

3.4.8 Differentiation: Sentence splitting, remove punctuation, remove parenthesis —— Remove punctuation, remove parenthesis

- The F-score

```
F1: 0.923409090909091 (micro), 0.9271653835124642 (macro), 0.9273189186008629 (weighted)
```

- Also similar as the previous experiment.

3.5 Conclusions

In this section it has been shown that the most potent step in preprocessing is sentence splitting for languages that do not use spaces. The reason for that is clear, the classifier considers the whole sentences as single words, and has a hard time discerning one language from the other, specially with Chinese, Japanese and Korean, which use very distinct linguistic signs from the other languages in the pool. What it is also worth mentioning, is that removing punctuation offers a slight improvement in performance, and the removal of the words between parenthesis (and quotation marks and others) does not show any improvement whatsoever. The best case in this stage has been **Differentiation: Sentence splitting —— Remove punctuation**. Comparing both PCA for No Preprocessing and the better strategy, we can appreciate an improvement in the separation between Chinese and Japanese, which was the most prominent error in classification in the No Preprocessing stage.

4 Conclusions

Along the project we have been experimenting with different parameters like classifier level, vocabulary size, machine learning algorithm and preprocessing methods. Additionally, we have conducted error analyses to understand specific patterns of misclassification, allowing us to refine the model iteratively.

We have found that that the best performance in the word level is directly proportional to the vocabulary size. Moreover, it seems like the Support Vector Machine is the best Machine Learning algorithm to classify the data, and the best preprocessing steps we have been in touch with are sentence splitting for syllabic languages (like Japanese or Chinese), and punctuation removal for the rest. However, since we detected some language switch mid-sentence in some cases, we expected that the removal of the words between special characters (parenthesis, quotation marks, etc) would improve the performance of the classifier, as the language switches happened mostly in these situation. Nevertheless, no improvement has been observed following this practise.

For future work, we will discuss potential changes for enhancing model training and performance:

- Improve the data quality. Correct the labels of the sentences.
- Train with more diverse examples from different languages.
- Lemmatization

A Appendix

Confusion Matrix & PCA Plots

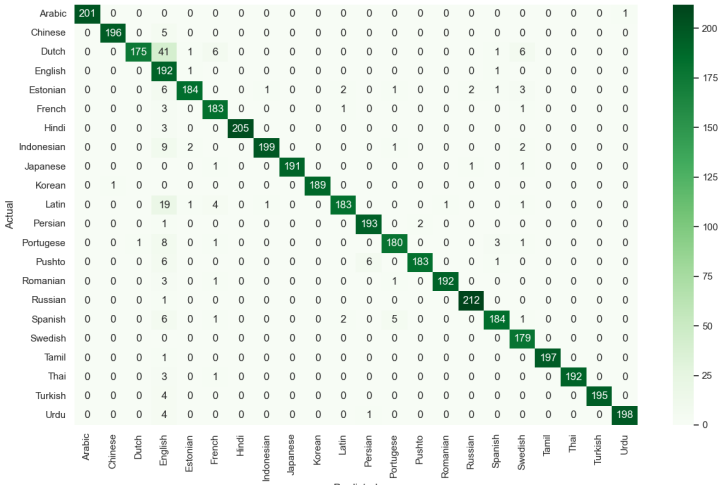

Figure 1: f(Char, 1000, Bayes, No Preprocess)

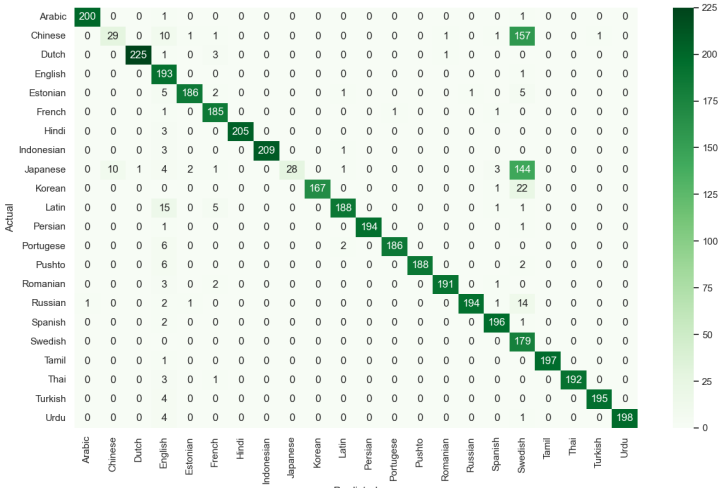

Figure 2: f(Word, 1000, Bayes, No Preprocess)

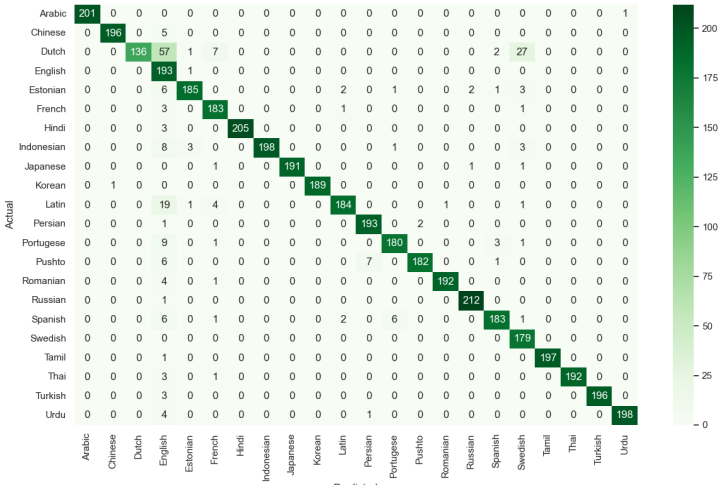

Figure 3: f(Char, 2000, Bayes, No Preprocess)

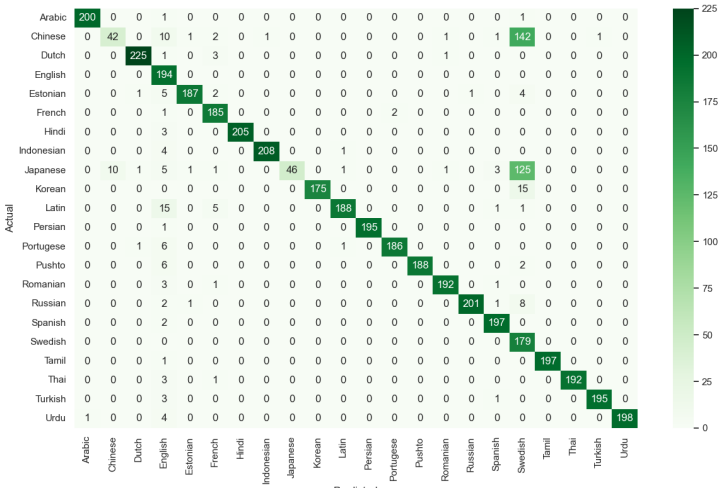

Figure 4: f(Word, 2000, Bayes, No Preprocess)

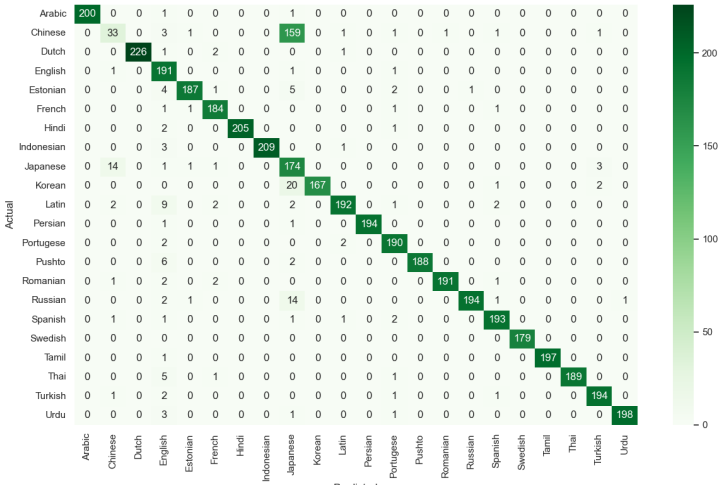

Figure 5: f(Word, 1000, SVM, No Preprocess)

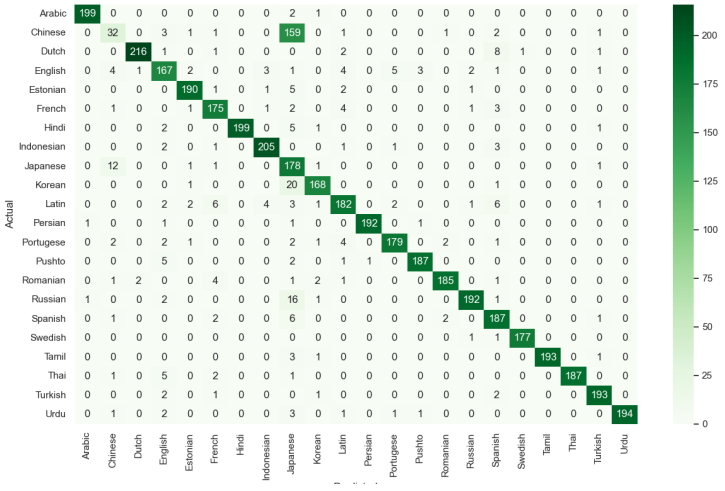

Figure 6: f(Word, 1000, Decision Trees, No Preprocess)

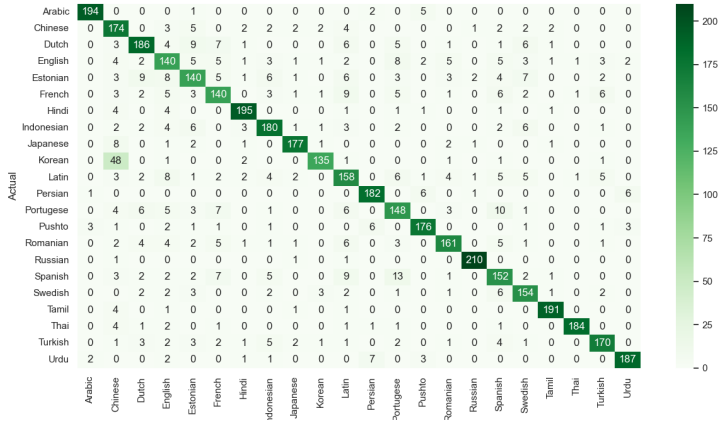

Figure 7: f(Word, 1000, Decision Trees, Sentence splitting, no differentiation)

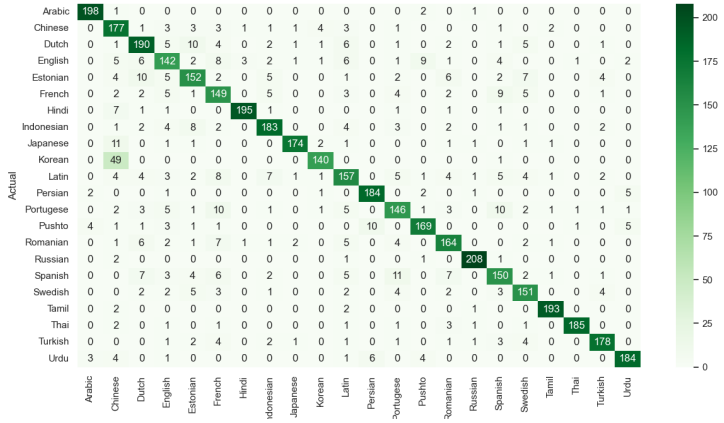

Figure 8: f(Word, 1000, Decision Trees, Sentence splitting, remove punctuation, no differentiation)

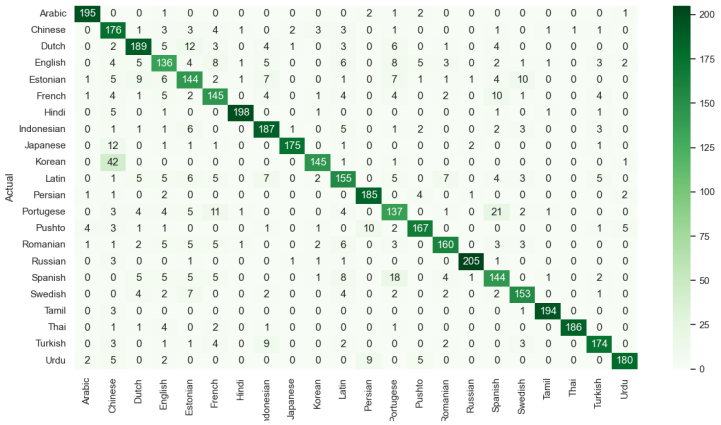

Figure 9: f(Word, 1000, Decision Trees, Sentence splitting, remove punctuation, remove parenthesis, no differentiation)

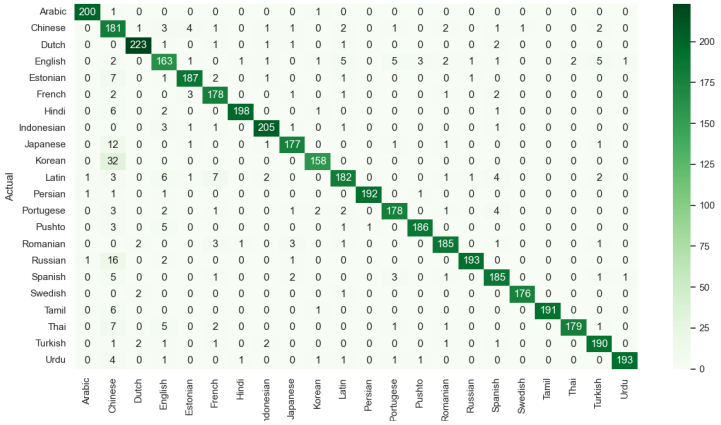

Figure 10: f(Word, 1000, Decision Trees, Differentiation: Sentence splitting —— nothing)

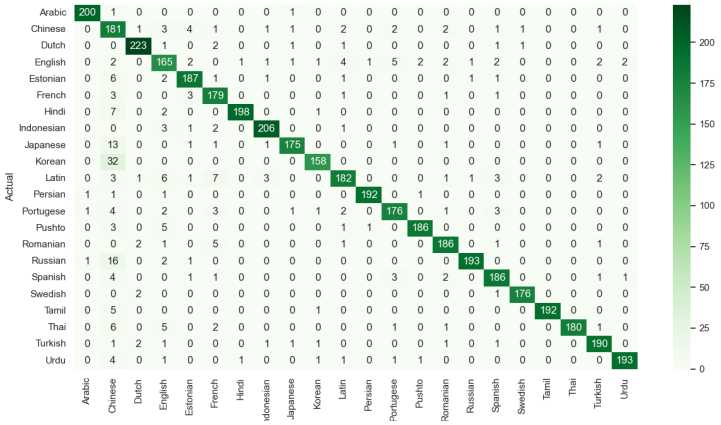

Figure 11: f(Word, 1000, Decision Trees, Differentiation: Sentence splitting —— Remove punctuation)

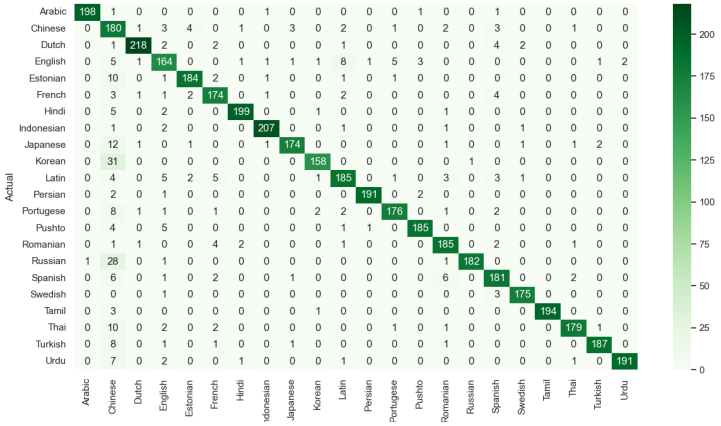

Figure 12: f(Word, 1000, Decision Trees, Differentiation: Sentence splitting —— Remove punctuation, remove parenthesis)

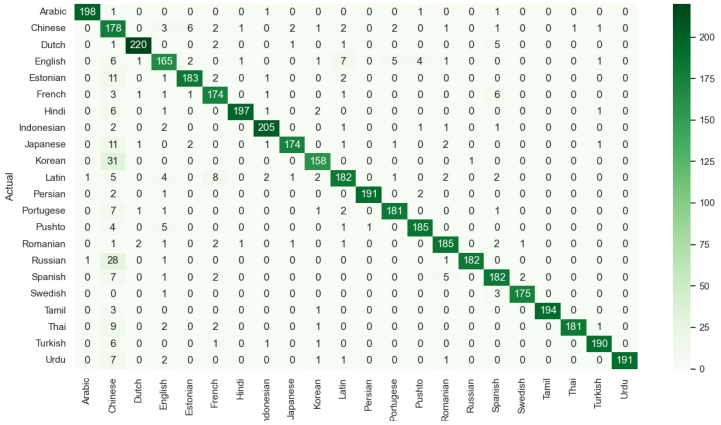

Figure 13: f(Word, 1000, Decision Trees, Differentiation: Sentence splitting, remove punctuation —— Remove punctuation, remove parenthesis)

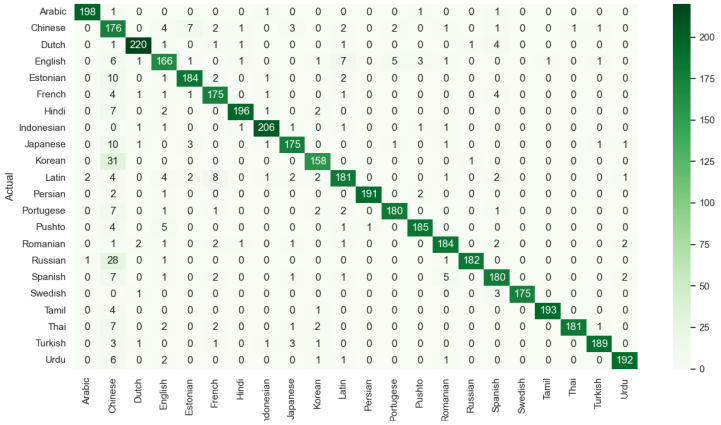

Figure 14: PCA(Word, 1000, Decision Trees, Differentiation: Sentence splitting, remove punctuation, remove parenthesis —— Remove punctuation, remove parenthesis)

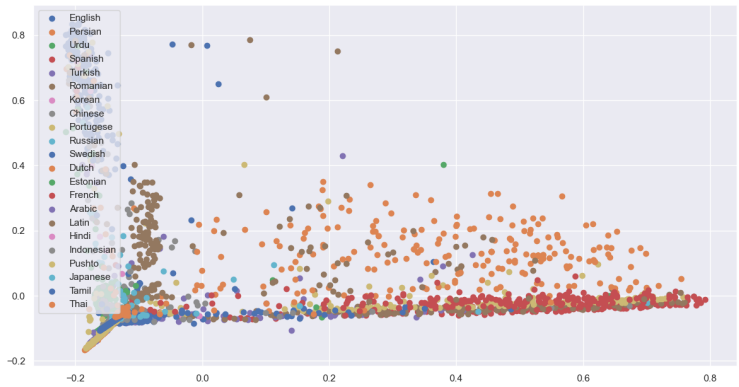

Figure 15: PCA(Word, 1000, Decision Trees, No Preprocessing)

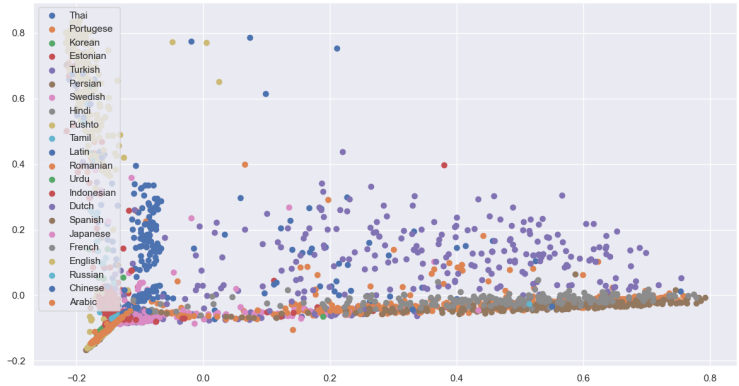

Figure 16: PCA(Word, 1000, Decision Trees, Differentiation: Sentence splitting —— Remove punctuation)