LANGUAGE IDENTIFICATION

Final Project Report

ABSTRACT

This project objective is Building a machine learning model to successfully identify the most popular languages. Many model have been tried and the results have been compared to find the best model.

Sara Elbesomy 201601849

NLP Course – CIE 553

Problem definition & Motivation:

There are more than 6500 language in the world. Definitely, not all of them are common but there are also many common languages, which may cause problems because they are very similar from a perspective of a non-speaker (Chinese and Japanese for example).

Moreover, language detection is a very necessary basic task before many NLP tasks such as document classification, machine translation and sentiment analysis. So it is obvious that language identification problem is very important and is worthy to be solved.

Dataset:

In this project, I will used a subset from WiLi-2018 Wikipedia dataset. The original dataset contains 235000 paragraphs of 235 languages. Each language in this dataset contains 1000 paragraphs.

Not all of the 235 languages are commonly used so, I just used a subset from the dataset that contains only 22 languages so, it include 22000 paragraphs. 1000 paragraphs for every language. A sample from the daatset is shown in figure (1). You can find the dataset here.

	Text	language
0	klement gottwaldi surnukeha palsameeriti ning ...	Estonian
1	sebes joseph pereira thomas på eng the jesuit...	Swedish
2	ถนนเจริญกรุง อักษรโรมัน thanon charoen krung เ...	Thai
3	விசாகப்பட்டினம் தமிழ்ச்சங்கத்தை இந்துப் பத்திர...	Tamil
4	de spons behoort tot het geslacht haliclona en...	Dutch

Figure (1)

The languages in the dataset and their frequencies are shown in figure (2).

```
Russian       1000
Swedish       1000
English       1000
Pushto        1000
Chinese       1000
Indonesian    1000
Dutch         1000
French        1000
Persian       1000
Japanese      1000
Turkish       1000
Urdu          1000
Estonian      1000
Thai          1000
Hindi         1000
Arabic        1000
Spanish       1000
Tamil         1000
Portugese     1000
Latin         1000
Korean        1000
Romanian      1000
```

Figure (2)

The most important advantage of this dataset is that it is balanced as shown in figure (3) so that I can use the accuracy metric in the evaluation of the models.

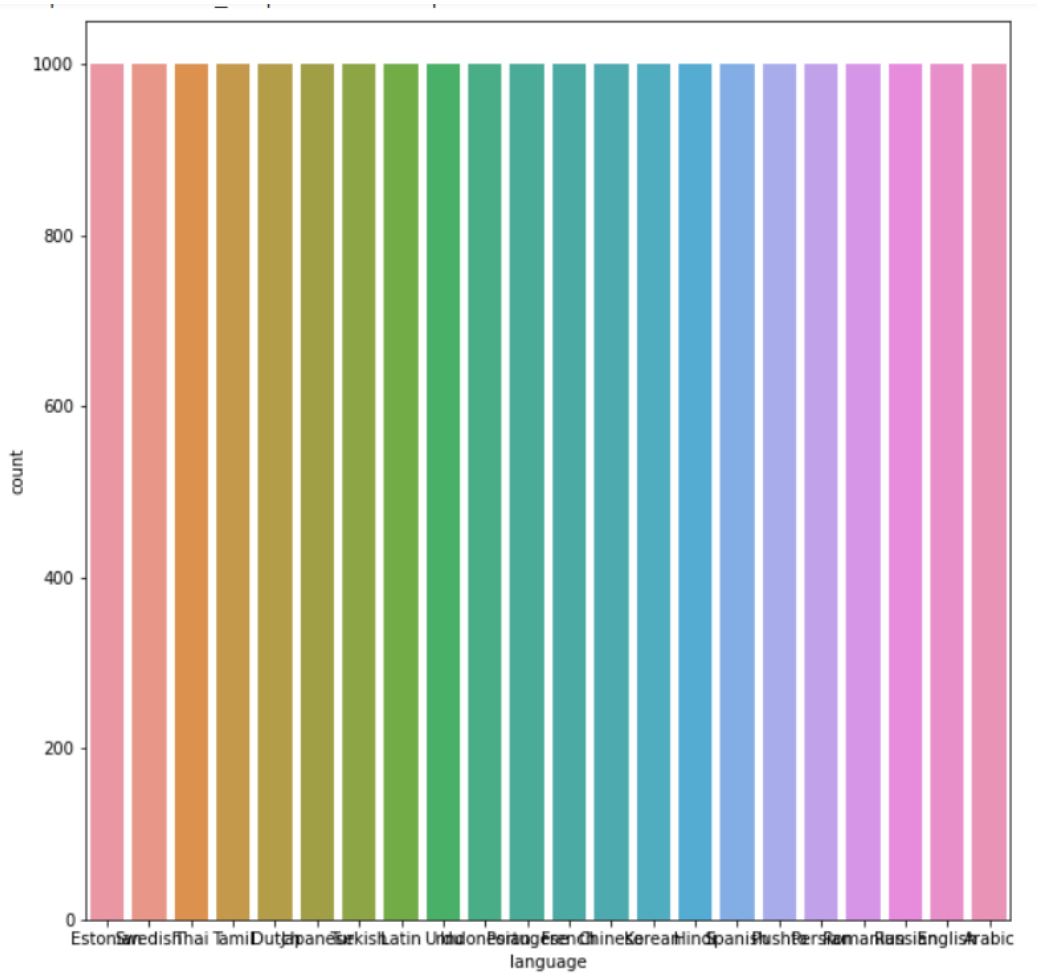

Figure (3)

Project Pipeline:

The objective of this project is comparing many machine learning models with each other's to reach the best one so, I used Logistic regression, support vector machine and a simple neural network specifically a multi-layers perceptron model.

Before using the models, we first need to preprocess the text data and represent it in a way that the models can deal with it and understand its features.

So, the project pipeline includes the following steps:

1. Cleaning the dataset if it is not clean.
2. Data preprocessing and vectorization.
3. Logistic regression model.
4. Support vector machine model.

5. Neural network model (Multi-layer perceptron).

Dataset cleaning:

I checked if there are missing values or none values in the dataset but I found it complete so, the data is clean.

Data preprocessing and vectorization:

Usually, any text data has to be preprocessed before feeding it to the models. The usual preprocessing steps are: tokenization, stop words removal, lower all the characters, punctuation removing, lemmatization and stemming.

Then preprocessing the text, we need to put it in numerical format to be understandable to the model. This could be achieved by representing it by vectors and there are many ways to do that.

What I have done in this project is to use two types of vectorizers, Bag of words and TF-IDF. Each one of them works in a different way but the goal is the same, representing the text with numerical vectors.

Bag of words builds these vectors by counting the frequency of each word in the corpus and then consider it as a feature, which represents the word.

TF-IDF works in a more advanced way. It also counts the frequency of each word in each document but it also compare it with the frequency of the same word in all the corpus so, we can notice to what extent this word is special in this document. At the end, it represents the text by vectors but these vectors has been built by a more complex way.

I used the library “sikitlearn” to import the “Countvectorizer” that could be considered “Bag of words” features extractor, and “tfidfvectorizer” that is obviously used “TF-IDF” features extractor.

The “Countvectorizer” function preprocessing the text by default. Exactly, it did tokenization, stop words removing, punctuation removing and lowering all the characters. These specific techniques are only what I need; I do not need stemming or lemmatization because I do not care about the meanings of the text while I am just interested in identifying the language. Therefore, there is no need to use “NLTK” library to make the preprocessing manually.

The “tfidfvectorizer” is also doing the preprocessing by default because it uses the “Countvectorizer” firstly so; there is also no need for manually preprocessing the text.

I used the following lines to build the two vectorizers as shown in figure (4):

```
1 vectorizer = feature_extraction.text.TfidfVectorizer(ngram_range=(1,3), analyzer='char')
2 vectorizer2 = feature_extraction.text.CountVectorizer(ngram_range=(1,3), analyzer='char')
```

Figure (4)

I used all the default values for preprocessing except for the “analyzer” value and the “ngram range”. For the analyzer, I chose it to be “char”, this means it will tokenize by character not by word so that all the analysis will be on characters not the complete word. I chose to do that because I have found a comparison between using the word analysis and using the character analysis in language detection models, and I found that the performance of using the character analysis is better as shown in figure (5).

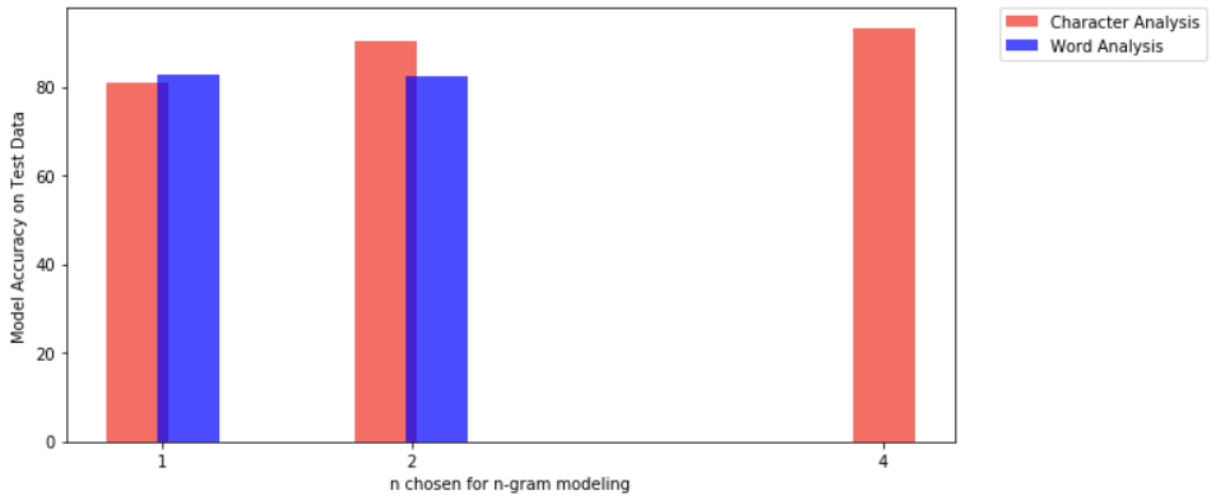

Figure (5)

As mentioned above, I used character analysis because it is better in the results but this was with classical machine learning models (Logistic regression and SVM in my experiments) but with neural networks model, I used word analysis.

The second value that I edited is the “ngram_range”. It means the range of the ngram that the vectorizer will include. I chose it to be from unigram to trigram.

Logistic regression model:

I used the “pipeline” module in “sikitlearn” library to make a pipeline that contains the vectorizer and the classifier as shown in Figure (6).

```
pipe_lr = pipeline.Pipeline([
    ('vectorizer', vectorizer),
    ('lr_clf', LogisticRegression())
])
```

Figure (6)

I tried the “logistic regression” model with the Bag of words vectorizer and with the TF-IDF vectorizer to compare the results and to know which victorizer is more suitable for this task.

The results of the accuracy are shown in the following table:

	Training accuracy	Test accuracy
LR with TF-IDF	98.5795 %	98.1818 %
LR with BoW	99.9943 %	98.4090 %

Table (1) – Logistic Regression Model

Results of Logistic Regression with TF-IDF:

As shown in figure (7) and figure (8), the confusion matrix and the classification report of the logistic regression model with TF-IDF vectorizer shows that the recall and the precision in almost 1 on most of the languages and this reflects the high accuracy. The wrong predicted instances are very few.

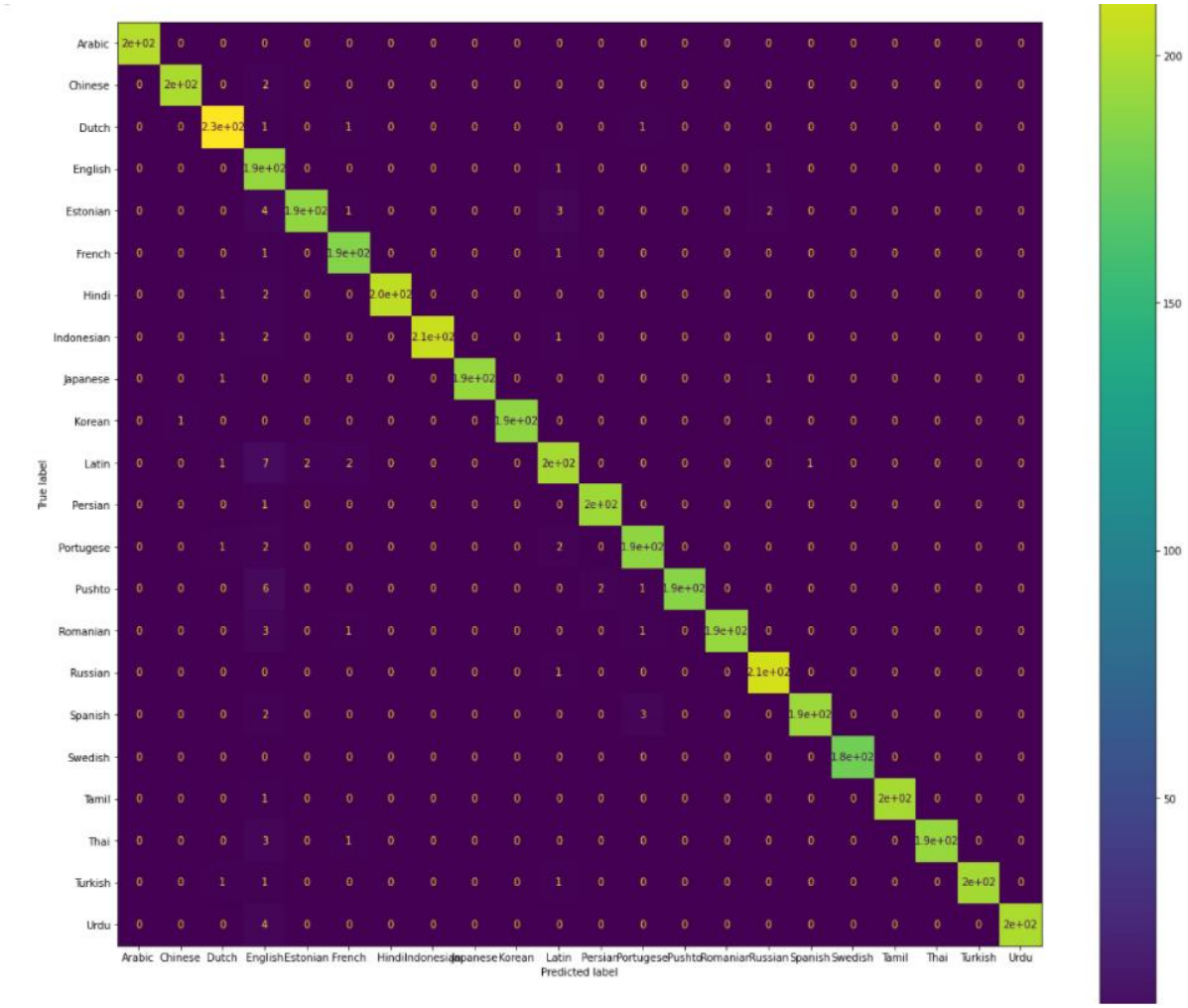

Figure (7) – Confusion matrix (LR with TF-IDF)

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	0.99	0.99	0.99	201
Dutch	0.97	0.99	0.98	230
English	0.82	0.99	0.90	194
Estonian	0.99	0.95	0.97	200
French	0.97	0.99	0.98	188
Hindi	1.00	0.99	0.99	208
Indonesian	1.00	0.98	0.99	213
Japanese	1.00	0.99	0.99	194
Korean	1.00	0.99	1.00	190
Latin	0.95	0.94	0.94	210
Persian	0.99	0.99	0.99	196
Portugese	0.97	0.97	0.97	194
Pushto	1.00	0.95	0.98	196
Romanian	1.00	0.97	0.99	197
Russian	0.98	1.00	0.99	213
Spanish	0.99	0.97	0.98	199
Swedish	1.00	1.00	1.00	179
Tamil	1.00	0.99	1.00	198
Thai	1.00	0.98	0.99	196
Turkish	1.00	0.98	0.99	199
Urdu	1.00	0.98	0.99	203
accuracy			0.98	4400
macro avg	0.98	0.98	0.98	4400
weighted avg	0.98	0.98	0.98	4400

Figure (8) – Classification report (LR with TF-IDF)

Results of Logistic Regression with BoW:

It seems that the results as shown in figure (9) and (10) when using the BoW with logistic regression is much better than using TF-IDF with the same model considering that the results of TF-IDF were excellent.

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	1.00	0.99	0.99	201
Dutch	1.00	0.98	0.99	230
English	0.90	0.96	0.93	194
Estonian	0.96	0.95	0.96	200
French	0.97	0.99	0.98	188
Hindi	1.00	0.99	0.99	208
Indonesian	0.99	0.99	0.99	213
Japanese	1.00	0.99	0.99	194
Korean	1.00	1.00	1.00	190
Latin	0.95	0.96	0.96	210
Persian	0.99	0.99	0.99	196
Portugese	0.97	0.98	0.98	194
Pushto	0.98	0.97	0.97	196
Romanian	0.99	0.97	0.98	197
Russian	0.99	1.00	0.99	213
Spanish	0.98	0.99	0.98	199
Swedish	1.00	0.99	1.00	179
Tamil	0.99	0.99	0.99	198
Thai	1.00	0.98	0.99	196
Turkish	1.00	0.99	0.99	199
Urdu	1.00	0.99	0.99	203
accuracy			0.98	4400
macro avg	0.98	0.98	0.98	4400
weighted avg	0.98	0.98	0.98	4400

Figure (9) – Classification report (LR with BoW)

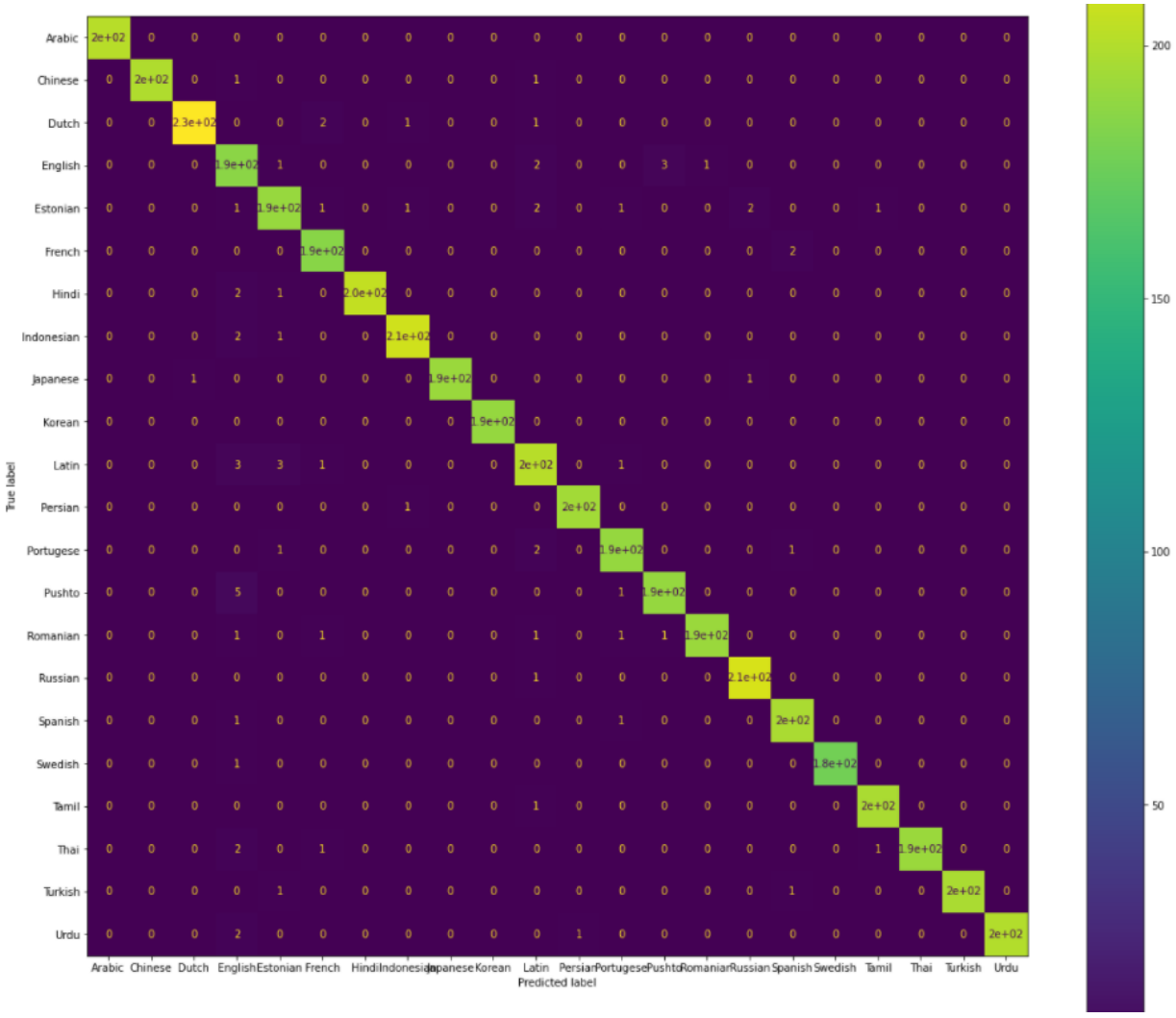

Figure (10) – Confusion matrix (LR with BoW)

Support Vector Machine Model:

I used the same function of "pipeline" to combine the vectorizer with the classifier; in this case, it is SVM. I also used the two victorizers with SVM to compare the results.

The results of the accuracy are shown in the following table:

	Training accuracy	Test accuracy
SVM with TF-IDF	99.2784 %	98.3863 %
SVM with BoW	97.9034 %	98.3863 %

Table (2) – Support Vector Machine model

Results of Support Vector Machine with TF-IDF:

As shown in figure (11) and figure (12), the results of SVM with TF-IDF is pretty excellent and it is very comparable to LR with BoW.

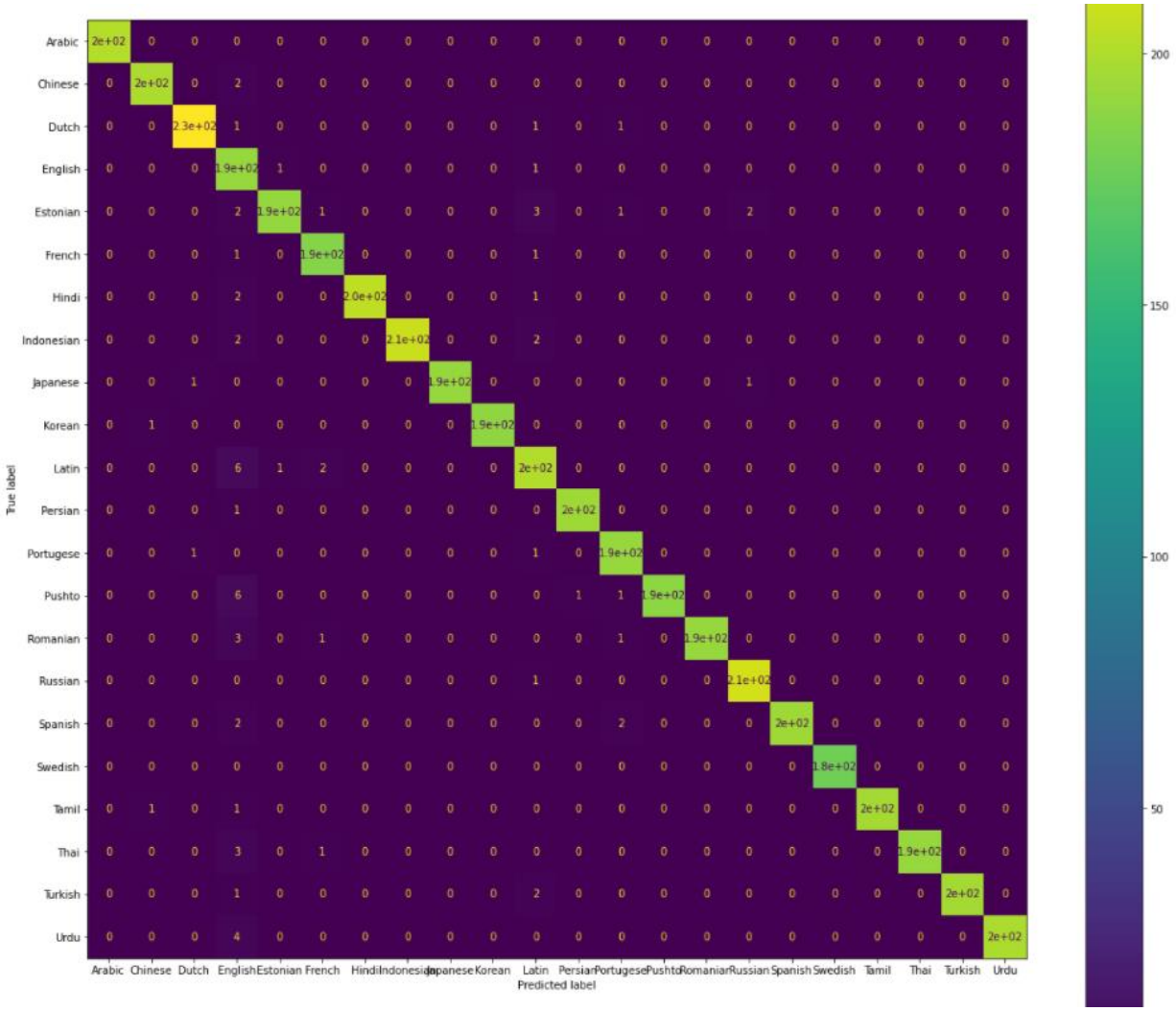

Figure (11) – Confusion matrix (SVM with TF-IDF)

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	0.99	0.99	0.99	201
Dutch	0.99	0.99	0.99	230
English	0.84	0.99	0.91	194
Estonian	0.99	0.95	0.97	200
French	0.97	0.99	0.98	188
Hindi	1.00	0.99	0.99	208
Indonesian	1.00	0.98	0.99	213
Japanese	1.00	0.99	0.99	194
Korean	1.00	0.99	1.00	190
Latin	0.94	0.96	0.95	210
Persian	0.99	0.99	0.99	196
Portugese	0.97	0.99	0.98	194
Pushto	1.00	0.96	0.98	196
Romanian	1.00	0.97	0.99	197
Russian	0.99	1.00	0.99	213
Spanish	1.00	0.98	0.99	199
Swedish	1.00	1.00	1.00	179
Tamil	1.00	0.99	0.99	198
Thai	1.00	0.98	0.99	196
Turkish	1.00	0.98	0.99	199
Urdu	1.00	0.98	0.99	203
accuracy			0.98	4400
macro avg	0.99	0.98	0.98	4400
weighted avg	0.99	0.98	0.98	4400

Figure (12) – Classification report (SVM with TF-IDF)

Results of Support Vector Machine with BoW:

As shown in the following figures (13) & (14), the results is not good as the previous model (SVM with TF-IDF) so that we can conclude that SVM is better with TF-IDF not BoW victorizer.

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	0.99	0.99	0.99	201
Dutch	0.96	0.97	0.97	230
English	0.79	0.98	0.88	194
Estonian	0.98	0.94	0.96	200
French	0.95	0.98	0.97	188
Hindi	1.00	0.99	0.99	208
Indonesian	1.00	0.97	0.99	213
Japanese	1.00	0.97	0.99	194
Korean	0.99	0.99	0.99	190
Latin	0.92	0.93	0.93	210
Persian	0.99	0.99	0.99	196
Portugese	0.96	0.95	0.96	194
Pushto	1.00	0.95	0.98	196
Romanian	1.00	0.97	0.98	197
Russian	0.99	1.00	0.99	213
Spanish	1.00	0.96	0.98	199
Swedish	1.00	1.00	1.00	179
Tamil	1.00	0.99	1.00	198
Thai	1.00	0.98	0.99	196
Turkish	1.00	0.98	0.99	199
Urdu	0.99	0.98	0.99	203
accuracy			0.98	4400
macro avg	0.98	0.98	0.98	4400
weighted avg	0.98	0.98	0.98	4400

Figure (13) – Classification report (SVM with BoW)

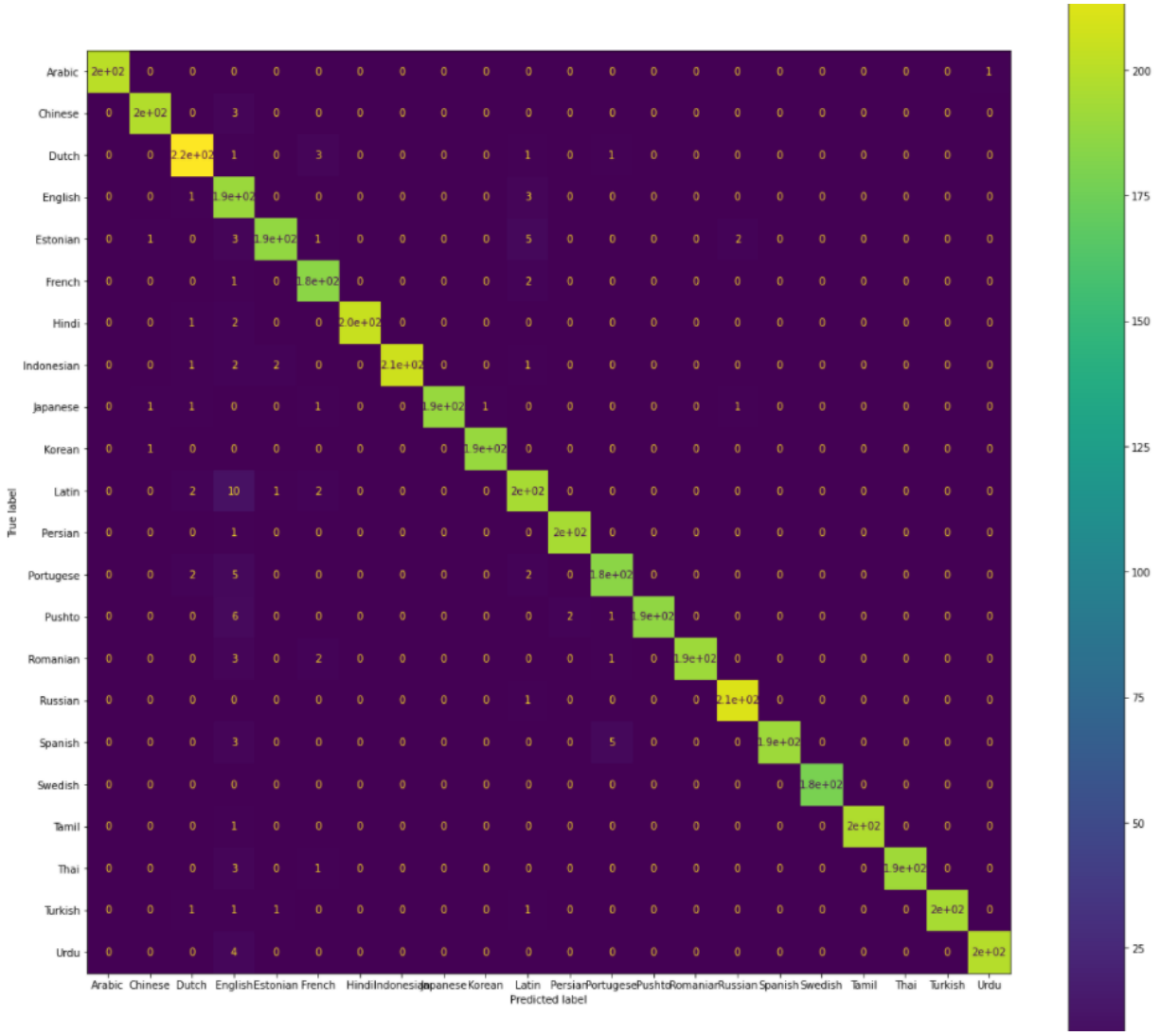

Figure (14) – Confusion matrix (SVM with BoW)

Neural network model (Multi-layer perceptron):

In this model, I also used the library "sikitlearn" to import the model "mlp". It is a very simple model with only three hidden layers with 20 neurons in each layer and the activation function for all the hidden layers is "Relu".

I tried the model with the two victorizers and here are the results:

	Training accuracy	Test accuracy
MLP with TF-IDF	99.9943 %	95.4318 %
MLP with BoW	99.9943 %	95.7954 %

Table (3) - Multi-layer perceptron model

Multi-layer perceptron with TF-IDF:

From figures (15) and (16), we can say that multi-layer perceptron model with TF-IDF is the worst model compared to the previous models. We can see that it has a big confusion between "Chinese " and "Japanese", Almost all of its wrong predictions due to a confusion between this two languages.

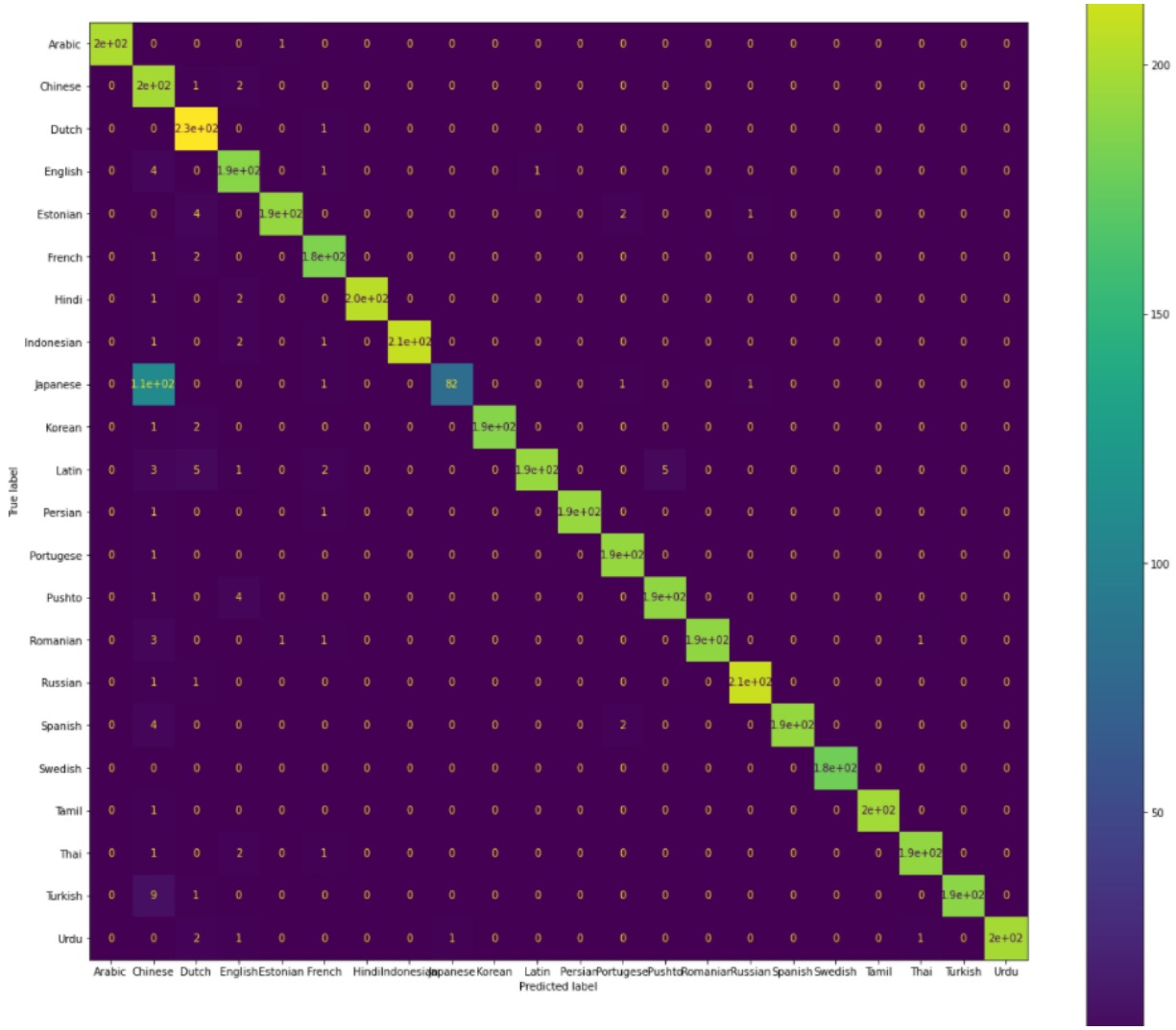

Figure (15) – Confusion matrix (MLP with TF-IDF)

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	0.58	0.99	0.73	201
Dutch	0.93	1.00	0.96	230
English	0.93	0.97	0.95	194
Estonian	0.99	0.96	0.98	200
French	0.95	0.98	0.97	188
Hindi	1.00	0.99	0.99	208
Indonesian	1.00	0.98	0.99	213
Japanese	0.99	0.42	0.59	194
Korean	1.00	0.98	0.99	190
Latin	0.99	0.92	0.96	210
Persian	1.00	0.99	0.99	196
Portugese	0.97	0.99	0.98	194
Pushto	0.97	0.97	0.97	196
Romanian	1.00	0.97	0.98	197
Russian	0.99	0.99	0.99	213
Spanish	1.00	0.97	0.98	199
Swedish	1.00	1.00	1.00	179
Tamil	1.00	0.99	1.00	198
Thai	0.99	0.98	0.98	196
Turkish	1.00	0.95	0.97	199
Urdu	1.00	0.98	0.99	203
accuracy			0.95	4400
macro avg	0.97	0.95	0.95	4400
weighted avg	0.97	0.95	0.95	4400

Figure (16) - Classification report (MLP with TF-IDF)

Multi-layer perceptron with BoW:

From the figures (17) and (18), we can notice that this model is also not working well compared with the previous models but it is better than MLP with TF-IDF and it has the same confusion between “Chinese” and “Japanese”.

	precision	recall	f1-score	support
Arabic	1.00	1.00	1.00	202
Chinese	0.89	0.65	0.75	201
Dutch	0.99	0.99	0.99	230
English	0.85	0.98	0.91	194
Estonian	1.00	0.96	0.98	200
French	0.96	0.99	0.97	188
Hindi	0.99	0.99	0.99	208
Indonesian	1.00	0.99	1.00	213
Japanese	0.71	0.89	0.79	194
Korean	0.99	0.99	0.99	190
Latin	0.99	0.94	0.97	210
Persian	1.00	0.99	0.99	196
Portugese	0.95	1.00	0.97	194
Pushto	1.00	0.96	0.98	196
Romanian	0.99	0.97	0.98	197
Russian	0.98	1.00	0.99	213
Spanish	0.98	0.98	0.98	199
Swedish	1.00	1.00	1.00	179
Tamil	1.00	1.00	1.00	198
Thai	0.99	0.98	0.99	196
Turkish	0.99	0.99	0.99	199
Urdu	1.00	0.98	0.99	203
accuracy			0.96	4400
macro avg	0.97	0.96	0.96	4400
weighted avg	0.97	0.96	0.96	4400

Figure (17) – Classification report (MLP with BoW)

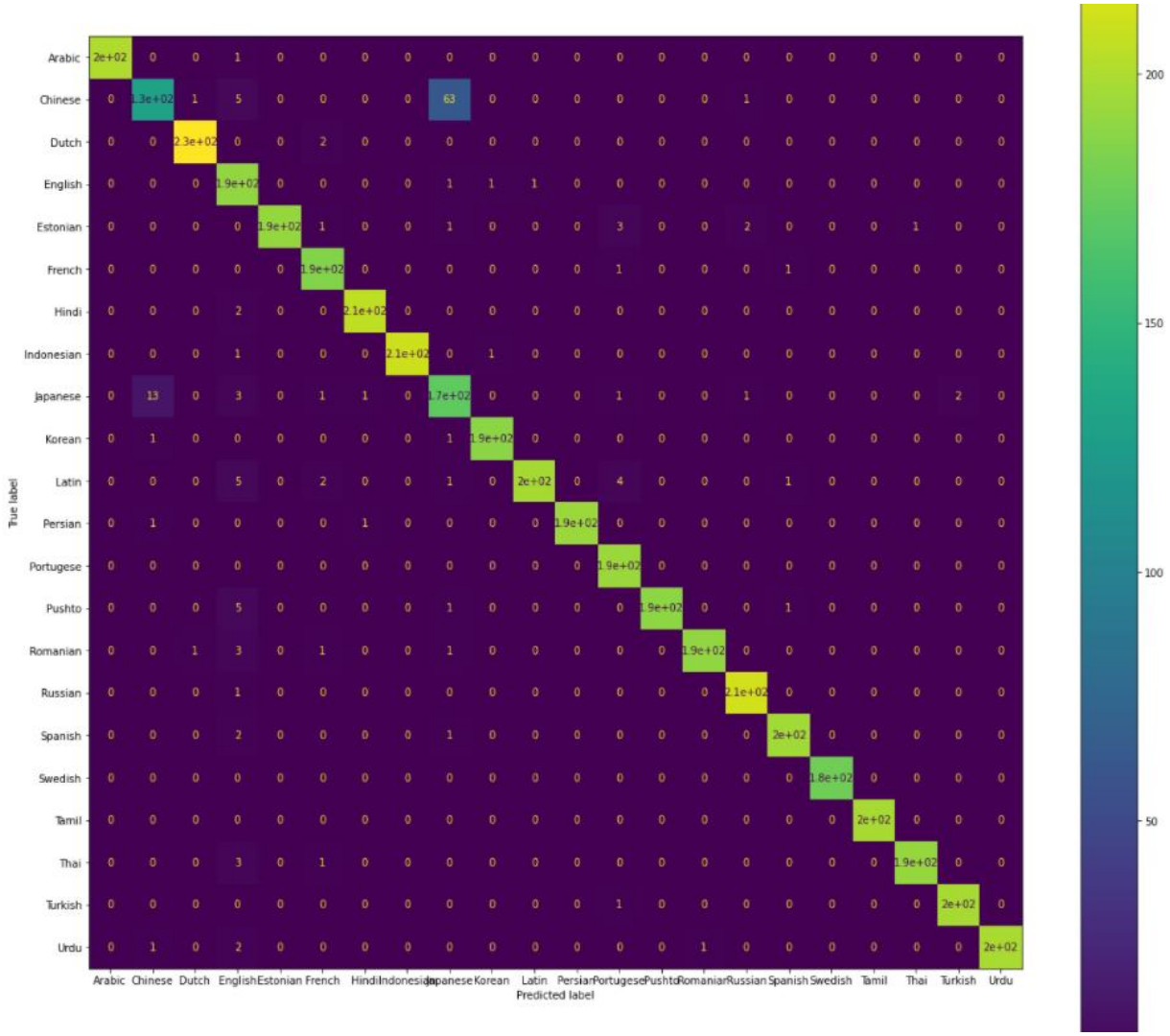

Figure (18) – Confusion matrix (MLP with BoW)

Analysis of the results:

	Training accuracy	Test accuracy
LR with TF-IDF	98.5795 %	98.1818 %
LR with BoW	99.9943 %	98.4090 %
SVM with TF-IDF	99.2784 %	98.3863 %
SVM with BoW	97.9034 %	98.3863 %
MLP with TF-IDF	99.9943 %	95.4318 %
MLP with BoW	99.9943 %	95.7954 %

Table (4) – All models comparison

General comments on the results (accuracies) and the confusion matrices and classification reports of all the models:

- It is obvious that the best model is the “Logistic regression” model with “Bag of words” victorizer.
- Genrally the results are very similar to each other and this indicates that the language identification task is not a complex one and it does not need high computational power or deep learning models.
- The performance of the neural network model (MLP) is not expected, at least for me, and I think that the problem is that it needs more experiments and fine tuning to choose the suitable number of hidden layers and the number of neurons in each layer.
- The most languages that caused a confusion to the model is “Chinese “ and “Japanese” and it is obvious that the reason behind this is that the two languages are used very similar writing systems.
- The unexpected thing is that “Arabic” and “Persian” did not cause the same problem while they are used the same characters.