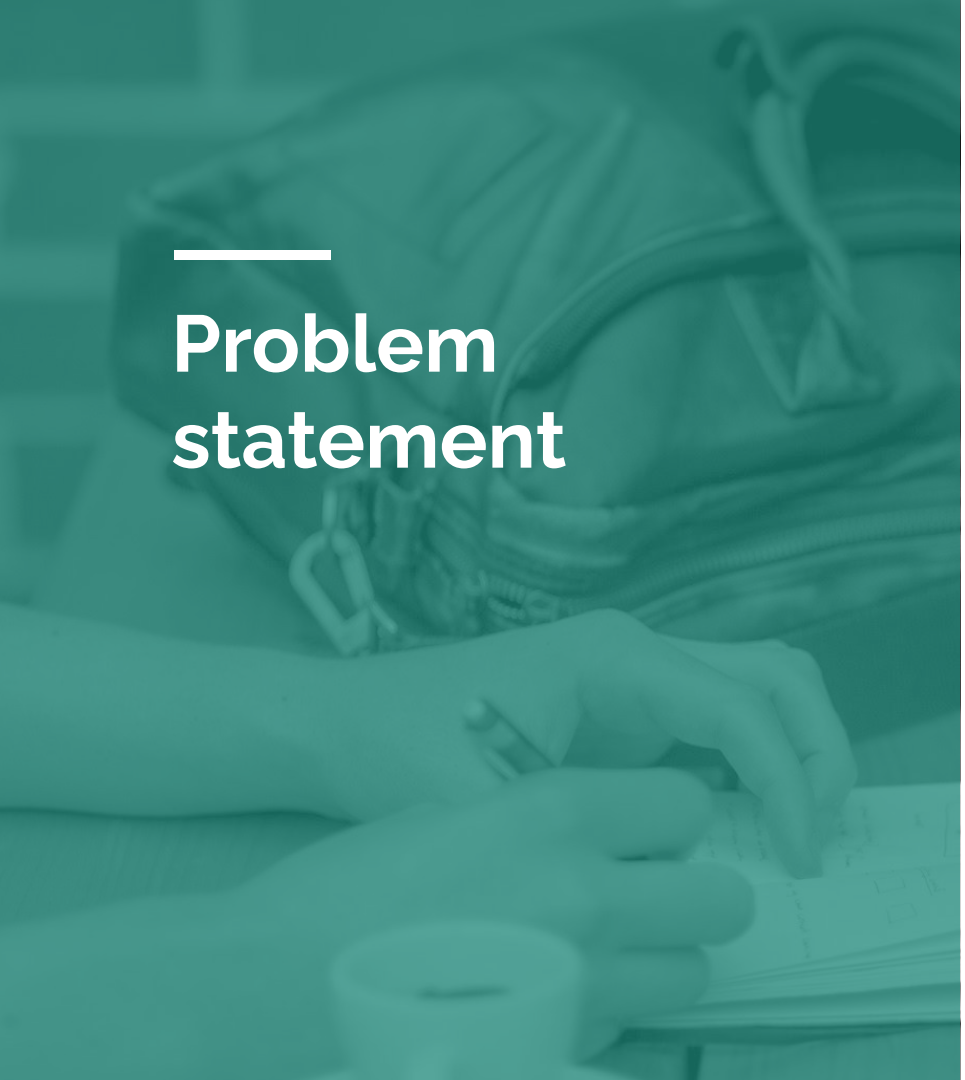
Language Identification Model

Group No : 13

Group Members :
Ankita Chandra(IIT2018053)
Ayushi Gupta (IIT2018118)
Puja Kumari (IIT2018191)

Introduction

- Language identification refers to the problem of detecting the language(s) in the textual document based on the script used for writing and observing the diacritics particular to a language.
- In our project we have presented a model to detect the language of a text accurately and efficiently.
- Language Identification typically acts as a pre-processing stage for both human listeners (i.e. call routing to a proper human operator) and machine systems (i.e. multilingual speech processing systems).



Problem statement

The focus of this project lies on the identification of efficient algorithms (in terms of memory requirements) by exploiting the knowledge about languages and their structures.

We are comparing results of Naive Bayes algorithm and KNN Algorithm.

What's new?

We are applying it on different feature set which consists of:

- Unigram Model
- Mixture(Uni+Bi) Top 1%
- Mixture(Uni+Bi) Top 50

Language Modelling Methods Available

The first stage is the modelling stage, where language models are generated. Such models consist of entities, representing specific characteristics of a language.The different approaches are:

- Unigrams
- Bigrams
- N Gram Based Approach:

N-Grams

- An N-gram is a combination of N tokens.
- N-gram is basically set of occurring words within given window .
- n=1 ,it is Unigram .
- n=2 , it is bigram .
- n=3 , it is trigram .
- Language model is generated from a corpus of documents using N-grams instead of complete words.

Unigrams

- It solely uses words up to a selected length to construct the language model .

The unigram language model makes the following assumptions:

- The probability of each word is independent of any words before it.
- Instead, it only depends on the fraction of time this word appears among all the words in the training text.

Bigrams

- A bigram is an n-gram for n=2 .
- A bigram or digram is a sequence of two adjacent elements from a string of tokens , which are typically letters, syllables, or words.
- Language models are generated based on specific amount of words, having the maximum frequency of all words occurring in a text or document.

Dataset Description

- The dataset we will be using is available on kaggle website. The dataset contains text information of about 22 different languages.
- The dataset has 22,000 rows and 2 columns.
- The different languages present in our dataset are as follows:

  'Arabic', 'Spanish', 'Swedish', 'English', 'Estonian', 'Tamil', 'Indonesian', 'Pushto', 'Dutch', 'French', 'Japanese', 'Turkish', 'Hindi', 'Latin', 'Persian', 'Romanian', 'Russian', 'Chinese', 'Portuguese', 'Korean', 'Urdu', 'Thai'

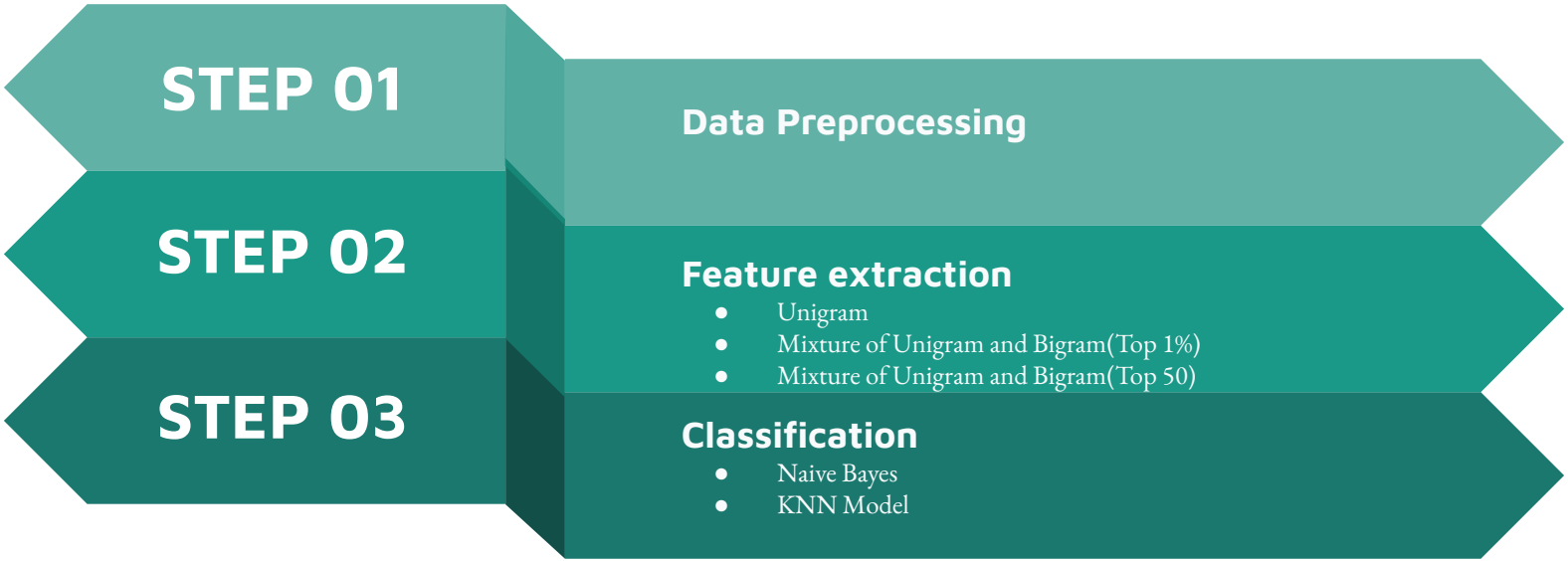
Requirements

The following python libraries will be required to make our model:

- Pandas
- Numpy
- Sklearn
- Matplot
- seaborn

We will use google colab for our project implementation.

PROPOSED METHODOLOGY



Data Preprocessing

- Data preprocessing makes our dataset suitable for the further steps .
- In this steps we remove any special symbols, punctuations ,extra space etc if present in our text as it is not required in the classification part.
- We then remove any numbers if present in our text,and we also remove any english letters if present in our text.

Feature Extraction

Unigram Model

The unigram model is also known as **the bag of words model**. Estimating the relative likelihood of different phrases is useful in many natural language processing applications, especially those that generate text as an output.

Using Uni-Grams allows us to identify *all* languages, which consist of *unique symbols*.

The Uni-Gram-approach might also be useful, if languages share *common symbols*, like the Latin symbols which are used in a lot of Indo-European languages including e.g. Romance (Italian, Romanian, Spanish, ...)

Unfortunately, using **Uni-Grams** we might face some issues, when we try to distinguish between languages having the same root, like for the Romance languages: Italian, Spanish, French, Portuguese, Catalan etc.

Why not Bigram?

The next larger piece of information are **Bi-Grams**. Unfortunately, using Bi-Grams might already blow up the required resources. So we should restrict ourselves here, to use only those, occurring most frequently.

But for the languages consisting on a few letters only, using Bi-Grams is very helpful. Since the most frequently used Bi-Grams differ a lot.

Whereas for Japanese and Chinese we can *not* find any Bi-Grams occuring at least in 1% of the cases.

Mixture of Unigram and Bigram

When we restrict ourselves to a limited number of features, it is important, that we will capture details for each language. Since Chinese consists of >3,000 of different symbols, the probability of the most frequently used Chinese Uni-Grams might be below the top 1000 used Bi-Grams of the other languages.

Using top 1%: We are using a lot of features per language, which might be already a good solution. *But choosing around 3079 features is already a lot*. And therefore the calculation is still expensive. It is possible to start the calculation based on around 3079 features so we delivered the results.

Taking top 50 per language:

We can restrict ourselves to the top 50 Uni- & Bi-Grams per language. This will lead to max of 22*50=1100 features. So, when dealing with the top-50-approach on these 22 languages, we will effectively use 564 features only. We built the data set for the models, based on our 564 features. From a theoretical perspective, it is most efficient to use a Mixture of the most common Uni-Grams and Bi-Grams.

Classification using Naive Bayes

- First we normalize the data.
- We have splitted data into 80:20 ratio for training and testing respectively
- After extracting the required features using Unigram Model, Mixture 1% model and Mixture top 50 model we will train our model using naive bayes algorithm.

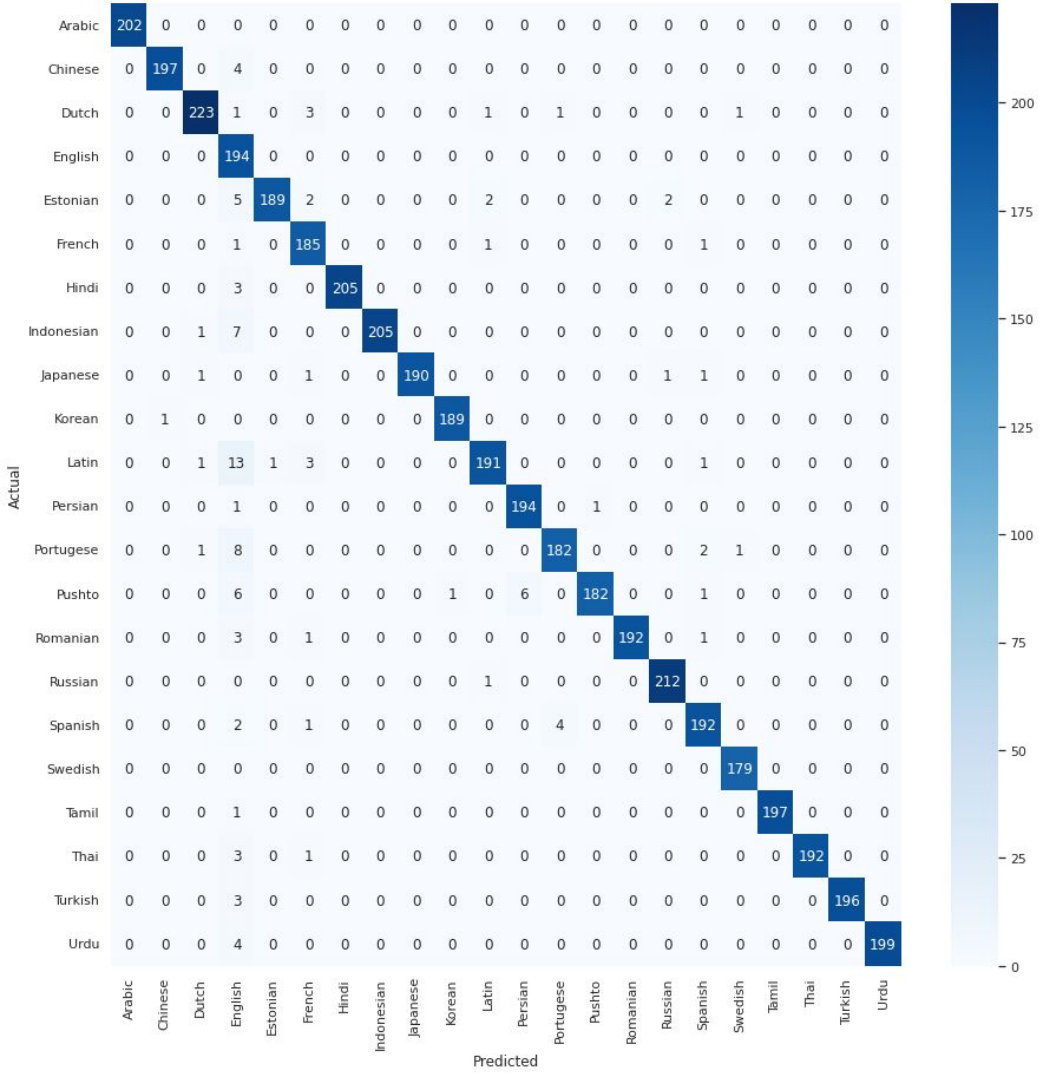
Classification using KNN model

- First we normalize the data.
- We have splitted data into 80:20 ratio for training and testing respectively
- After extracting the required features using Unigram Model, Mixture 1% model and Mixture top 50 model, we will train our model using KNN algorithm.
- We are taking the value of k=5 which is default

Results

Below is the ranking according to the weighted F1 scores we obtained

1. 0.9751510056992273 **NB** on **Mix Top 50**
2. 0.9746025107937131 **NB** on **Mix Top 1%**
3. 0.9739276244435526 **kNN** on **Mix Top 1%**
4. 0.972350612063613 **kNN** on **Mix Top 50**
5. 0.9545028869296119 **kNN** on **Unigram**
6. 0.9201996483569281 **NB** on **Unigram**



We obtained the confusion matrix of Naive Bayes classification model using top 50 dataset of the mixture of Unigram and bigram model.

Conclusion

We saw how we trained using dataset extracted using Unigram, mixture of 1% and mixture of top 50 on Naive Bayes and kNN model . The best accuracy we got was from mixture of top 50 model using naive bayes.

References :

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9399594
2. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.414.1623&rep=rep1&type=pdf
3. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.362.2541&rep=rep1&type=pdf
4. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.682.1583&rep=rep1&type=pdf

THANK YOU!