# EDAN20 Assignment 3
# Language classifier

Matilda Andersson, ma4748an-s

September 2020

## 1 Introduction

The purpose of this laboratory report is to describe and comment on the language detector composed in this assignment. The language detector used neural networks from the Python sklearn module, with a hidden layer of 50 nodes and a rectified linear activation function. The language data was collected from Tatoeba and consisted of more than 8 million short texts in 347 languages.

## 2 Process description

The following subsections contain a more in depth description of the process for creating the language classifier.

### 2.1 Preparing the data

For this assignment only the three languages Swedish, English and French were to be used for the model, which induced us starting the data preparation with filtering out all texts written in one of these languages. In the following step, the character ngrams for each sentence were computed, resulting in each sentence receiving additional features consisting of its character unigram, bigram and trigram count. The ngrams used relative frequency and were presented in a Counter dictionary. Unfortunately, this step in the process were highly burdensome for the computers memory and caused the program to be killed by the operative system due to exceeding memory usage. In order to solve this issue, we proceeded with only the unigram count as an additional sentence feature.

### 2.2 Building the X and Y matrices

To create the X matrix needed for the model creation, it was necessary for us to transform the dictionaries corresponding to each sentence, into numerical vectors. Using a class from the scikit-learn library, namely the DictVectorizer and its method `fit_transform()`, we were able to convert the dictionaries into the numerical vectors desired.

In order to obtain the Y vector in the required format, we created two conversion tables enabling us to refer to the different languages using an index, and numbers, instead of letters. This is needed as the networks only handles numerical values and not language as we now it.

### 2.3 Building the model

As mentioned in the introduction, a neural network from the sklearn libraray were used. Creating a MLPClassifier object with 50 hidden layers, which did 5 iterations and used a rectified linear activation function, we then went on to splitting the data into a training set and one validation set. Training the model with five iterations resulted in a loss of 0.05931398 on the last iteration, meaning that the loss function did not get the chance to converge. As an improvement of the model, the number of iterations could be increased until the loss function converges.

### 2.4 Predicting

Using our validation data we ran a test prediction through the model and received an accuracy score of 0.97874, with the following classification report seen in figure 1:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| swe | 0.97 | 0.89 | 0.93 | 7586 |
| eng | 0.98 | 0.99 | 0.99 | 271268 |
| fra | 0.96 | 0.96 | 0.96 | 88083 |
| accuracy | | | 0.98 | 366937 |
| macro avg | 0.97 | 0.94 | 0.96 | 366937 |
| weighted avg | 0.98 | 0.98 | 0.98 | 366937 |

Micro F1: 0.9787402197107405
Macro F1 0.9588481097653005

Figure 1: Table showing the classification result after five iterations.

Finally, we tested our prediction model on the following sentences: *"Salut les gars !", "Hejsan grabbar!", "Hello guys!" and "Hejsan tjejer!"* and recived the following language predictions `['fra', 'swe', 'eng', 'swe']`. Even though our model was only trained using unigrams as feature lables and with a non converging number of iterations, it managed to correctly predict the languages. In order to improve the model we could though let the fitting and training part run more iterations and the training data could also be increased in feature sizes to include also bigrams and trigrams.

## 3 Additional tasks

### 3.1 Google's Compact language detector

As an additional duty, we were tasked to study Google's language detector, CLD3, which is a neural network created for language prediction using the count of character ngrams from the input text in order to computes the relative fraction of times each of them appears in the text. Hashing down the ngrams to an id within a small range, each id gets represented by an embedding vector. The model consists of an embedding layer, a hidden rectified linear layer and a softmax layer and passing the input text through these layers a prediction can then be made.

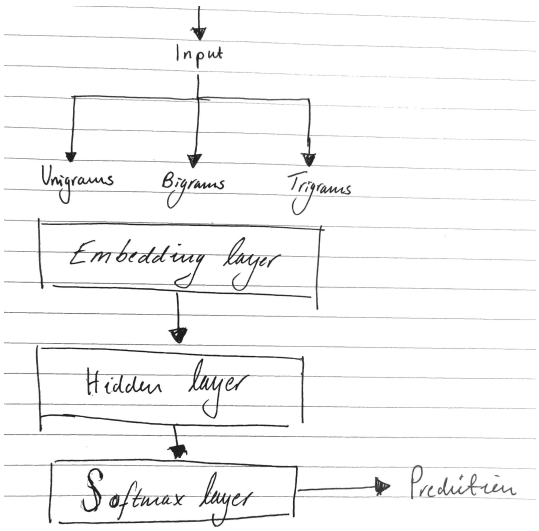


Figure 2: Outline of the CLD3 overall architecture.

### 3.2 Understanding the X matrix

Furthermore, in order to fully understand the X matrix we manually filled in the number of occurrences for the uni- and bigrams 'a', 'b', 'n', 'an', 'ba', 'na' in the sentences presented in the beginning of the assignment. The resulting matrix is presented in figure 3.

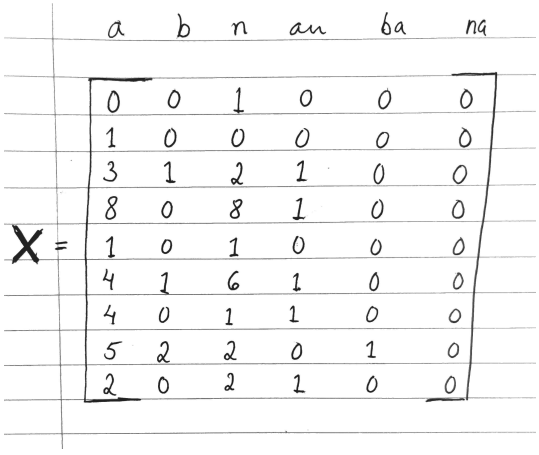


Figure 3: Table showing the classification result after five iterations.