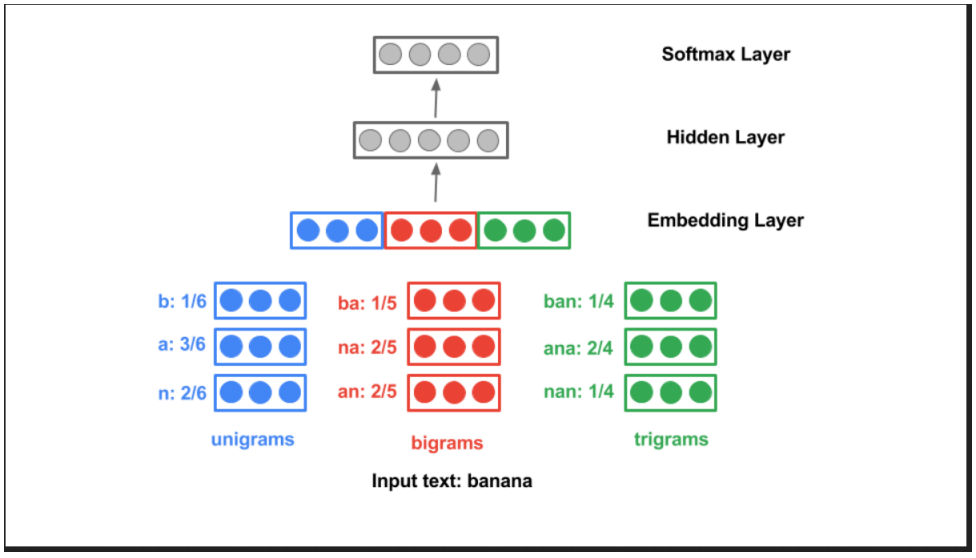
EDAN20 - Assignment 3

Oliver Cosic

September 2021

1 Introduction

In this assignment we try to mimic CDL3 to predict languages of a couple of sentences. CLD3 is a neural network model for language detection. It uses ngrams from the input texts and calculates the fraction of times each of them appears in the text analyzed. This is what we do in the first part of the assignment. The model then continues to average the embeddings that corresponds to each ngram and their respective fractions. The averaged embeddings are concatenated to produce the embedding layer. The remaining components are a hidden layer and a softmax layer. This we do in the second and third part of the assignment. And then to predict you simply performr a forward pass through the network, as we do in the last part of the assignment. All of these steps are demonstrated in the figure below with the example of the word banana.



* Summarize CLD3 and outline its architecture * Identify the features used by CLD3 * Include the feature matrix you computed manually

2 Extracting the features

We are working with a huge dataset, but for this assignment we settle with three langauges:Swedish, English and French. So we extract all the sentences with these languages in following way: We then wrote three functions counting

```
In [4]: dataset_small=[]
        for item in dataset:
            if item[1]=='eng' or item[1]=='fra' or item[1]=='swe':
                dataset_small.append(item)
```

unigrams, bigrams and trigrams and returning the relative frequencies. Below you can see how we did that for unigrams. The last thing we need to do in this

```
In [6]: def count_chars(string, lc=True):
            count_chars={}
            if lc==True:
                string=string.lower()
            splittedString=[char for char in string]
            total=len(splittedString)
            for char in splittedString:
                if char in count_chars.keys():
                    count_chars[char]+=1
                else:
                    count_chars[char]=1
            for key in count_chars.keys():
                count_chars[key]=count_chars[key]/total

            return count_chars
```

part is to extract features from our dataset and this was done in the following way:

```
: dataset_small_feat=[]
  for item in dataset_small:
      currentString=item[2]
      uni=count_chars(currentString)
      big=count_bigrams(currentString)
      tri=count_trigrams(currentString)
      feat=(item[0], item[1], item[2], uni, big, tri)
      dataset_small_feat.append(feat)
```

3 Building X and Y

Building X we only use unigrams since its enough for this particular assignment and makes the process fast and more memory efficient. So we extract the unigrams and their relative frequencys as shown below: And the to convert the X

```
listedDicts=[]
for item in dataset_small_feat:
    listedDicts.append(item[3])
X_cat=listedDicts
```

cat matrix to numerical representation we use scikit learns DictVectorizer and fit transform .

Building Y we extract the language symbols in a very similar way as we did with X. We also convert this matrix into a numerical representation where each langauge has a specific number. In my case french:0, english:1, swedish:2.

4 Building the model

We built the neural network as seen below: Then we split our dataset into a

```
clf = MLPClassifier(hidden_layer_sizes=(50, ), max_iter=5, verbose=True)
```

```
clf
```

```
MLPClassifier(hidden_layer_sizes=(50,), max_iter=5, verbose=True)
```

training set and a validation set and used the training sets to train the model. We used the validation set to test the predictions and confirmed the model is working.

5 Predicting

The last part was to predict what languages the following sentences are: We

```
docs = ["Salut les gars !", "Hejsan grabbar!", "Hello guys!", "Hejsan tjejer!"]
```

created features from the list of sentences. The feature matrix is shown here: And then we predicted the languages and got the following results:

```
X_test
```

```
array([[0.1875    , 0.0625    , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.06666667, 0.06666667, 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.09090909, 0.09090909, 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.07142857, 0.07142857, 0.        , ..., 0.        , 0.        ,
        0.        ]])
```

```
pred=clf.predict(X_test)
```

```
pred_languages=[]

for item in pred:
        pred_languages.append(y_symbols[item])
pred_languages
```

```
['fra', 'swe', 'eng', 'swe']
```