EDAN20 - Lab 3

André Frisk - an8218fr-s

September 2021

Language Detector

This lab is about implementing a functional language detector. When sending in a sentence to the program the language detector determines what kind of language it is. This is done using a implementation inspired by Googles compact language detector (CLD3). To make this work we use DictVectorizer and MLPClassifier from *sklearn* to make this program work. This lab focuses on the English, French and Swedish dictionary which will be a huge dataset downloaded from Tatoeba.

What is DictVectorizer and MLPClassifier?

The functions used in these libraries will be explained later.

- DictVectorizer: Transforms lists of feature-value mappings to vectors. This transformer turns lists of mappings (dict-like objects) of feature names to feature values into Numpy arrays or scipy.sparse matrices for use with scikit-learn estimators.
- MLPClassifier: Stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Relies on an underlying Neural Network to perform the task of classification. This model optimizes the log-loss function using LBFGS or stochastic gradient descent.

Characters Ngrams

The information of the dictionaries of the English, French and Swedish dictionary is stored in a dataset called `dataset_small_Feat`. The dataset is structured that every line in the set gives the textid, which language it is, the text and the character counts. For character counting we count by unigram, bigram and trigram which is one, two and three letters together respectively. When doing this we calculate the relative frequency of that unigram/bigram/trigram of the total words. This is useful later.

Building X

Now a matrix called **X** will be built using only the unigrams to speed up the process. First a new list of datapoints from `dataset_small_feat` is created where every item is a dictionary of the unigram probability of the letter/word. Then this list is used with the `fit_transform()` function from the DictVectorizer class which scales the training data and also learns the scaling parameters of that data, this model will learn the mean and variance of the features of this data which then is used to scale our test data.

Building y

In the `y` vector we need the language symbols from `dataset_small_feat`. The languages are extracted and the languages gain a numerical value which is inserted in a vector. The languages corresponds to these values: English = 0, French = 1, Swedish = 2.

Building the model and prediction

In the last part we use a neural network with a hidden layer (a layer in between input layers and output layers) of 50 nodes using MLPClassifier with a maximum of five iterations. Firstly the MLPClassifier (called `clf` for now) uses two training sets from **X** and `y` where we use random indices to split the dataset. We use these two training sets with `clf.fit()` in these five iterations which calculates the parameters and saves them as an preparation for later. The Loss function prints out the prediction error for every iteration. Then the `clf` will predict the languages in the validation sets (predicts the target values of the reserved test datasets) and checks if the accuracy is high (which it should be at his point, otherwise the program will not work as intended).

Now comes the part of where the program shows what it can do. We have a vector with four different sentences which the program will try to predict which language it corresponds to. These sentences is sent in to the `count_chars()` function sends out a list of the word probability for every sentence which will be used with `DictVectorizer.transform()` which applies the previous scaling to these dictionaries in the list (which is inserted in a feature vector). This feature vector is then used to predict with the `clf`. If the programming have been done correctly then the program should print out from the prediction: ['fra', 'swe', 'eng', 'swe'] which indicates which sentence corresponds to which language.

CLD3

CLD3 is a neural network model for language identification. Firstly the program extracts character ngrams from the input text and computes the fraction of times they appear in the text. The model averages the embeddings corresponding to each ngrams type according to the fractions. These average embeddings are then concatenated to produce the embedding layer (a structure of the word). Then there exist a hidden layer and softmax layer. The softmax layer assigns decimal probabilities to each word in a multi-word problem which can be used to find the correct word.

Here is a matrix applied with the CLD3 technology used with indices from the first three sentences from the English, French and Swedish languages (Col 1, Row 4: The first one will be 5 if the letter 'à' counts as the letter a).

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 2 & 1 & 0 & 0 \\ 8 & 0 & 8 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 4 & 1 & 5 & 1 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 0 \\ 5 & 2 & 2 & 0 & 1 & 0 \\ 2 & 0 & 2 & 1 & 0 & 0 \end{bmatrix} ; \mathbf{y} = \begin{bmatrix} \text{eng} \\ \text{eng} \\ \text{eng} \\ \text{fra} \\ \text{fra} \\ \text{fra} \\ \text{swe} \\ \text{swe} \\ \text{swe} \end{bmatrix}$$

Following is the architecture of the CLD3 model:

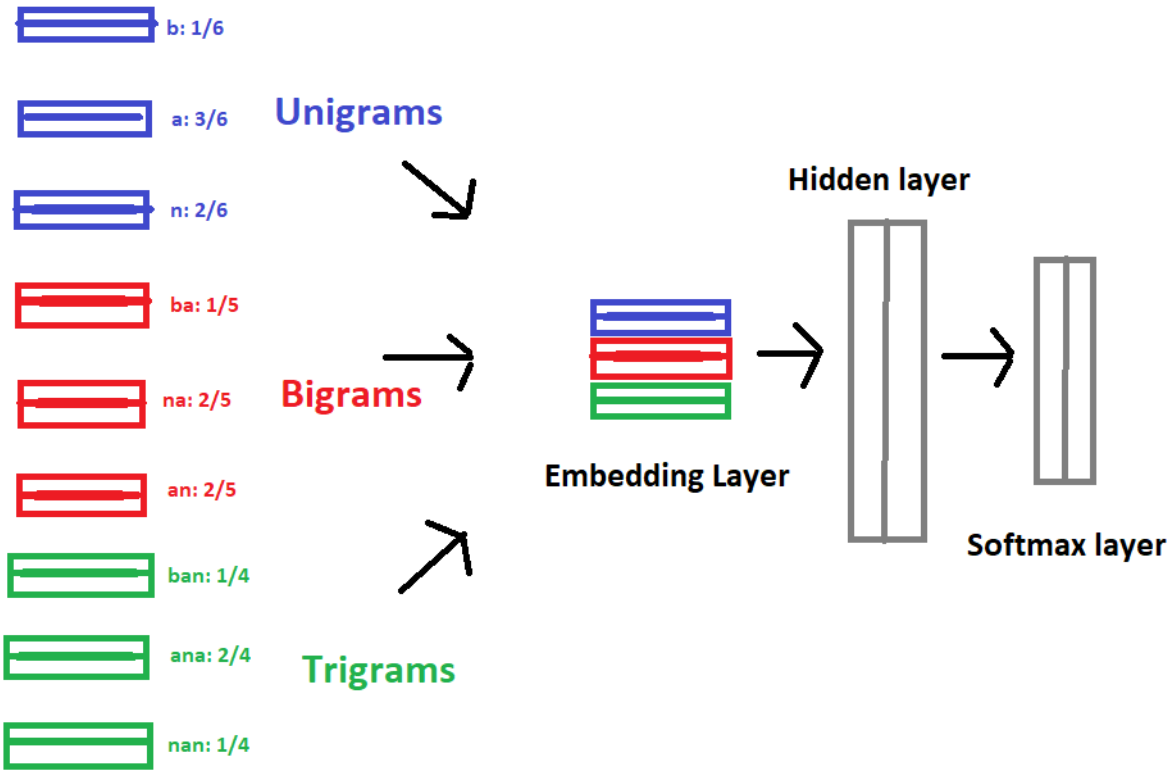

Figure 1: Architecture modeled by the student for CLD3