Artificial Intelligence, University of Bucharest

Practical Machine Learning

Project 1: “Geo-location of German Tweets” Kaggle competition

Student: Sotir Anca-Nicoleta (group 407)

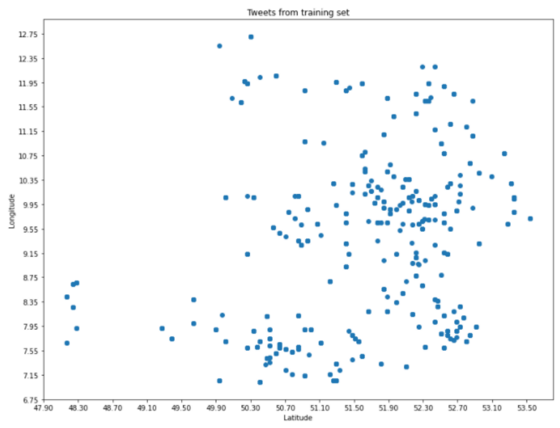
In this documentation will be presented my approaches to predict the geographic location (latitude and longitude coordinates) of German tweets. In addition to the two regression models presented, two different feature extraction methods were used. More concretely, the first approach consists of extracting word vectors using spaCy model for German written text and applying SVR; the second approach consists of using a simple neural network with an embedding layer.

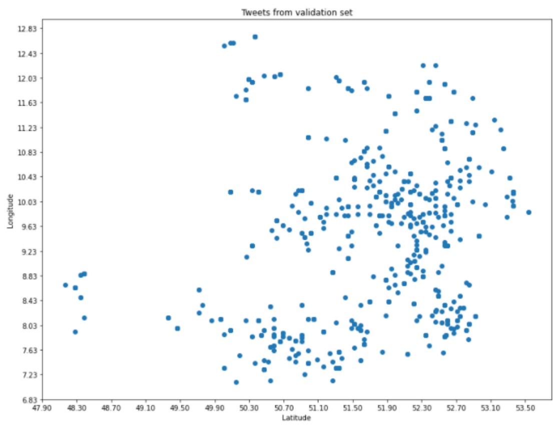
My final result for this competition is a mean absolute error of 0.59955, ranking on the final competition leaderboard on the 54th place. The two submissions chosen for the final leaderboard were the neural network and a combined result of the neural network and SVR methods.

Submission	Private Score	Public Score
Neural network	0.59454	0.59330
Neural network (training also on validation data)	0.59955	0.57823
Combined results of the neural network and SVR model	0.60145	0.59297
SVR (training also on validation data)	0.68154	0.68084
SVR	0.68507	0.68394
Random forest (Regression trees)	0.77688	0.76062

These are my submissions and their corresponding results.
Marked with green is the submission which corresponds to the final leaderboard ranking.

Data visualization

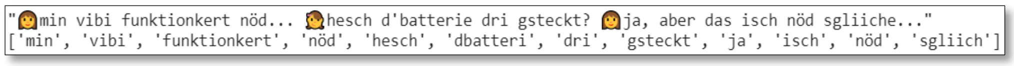




I plotted the coordinates for both training and validation datasets.
It can be seen that they have similar distributions.

Text preprocessing

I have chosen to preprocess the text prior to feature extraction. This process contains of the following steps: removing emoticons and emoji (using the emot package), lowercasing, removing punctuation, discarding German stop words and, finally, word stemming (using the nltk package).



```
"👩min vibi funktionkert nöd... 👧hesch d'batterie dri gsteckt? 👩ja, aber das isch nöd sgliiche..."
['min', 'vibi', 'funktionkert', 'nöd', 'hesch', 'dbatteri', 'dri', 'gsteckt', 'ja', 'isch', 'nöd', 'sgliich']
```

This is the first entry of the test set before and after text preprocessing.

The first approach – word vectors and Support Vector Regression

After preprocessing, the text is given to a spaCy pre-trained model for German written text (news and media)[1]. This allows further feature extraction, a 300 dimension vector being assigned to each word. I decided to encode each tweet in a vector of dimension 300 which is the mean of its word vectors. The encoded tweets are then scaled using a **StandardScaler** and given to a **Support Vector Regression** model wrapped in a **MultiOutputRegressor** (using sklearn). The **rbf** kernel had better results than the linear kernel. I also tried different values for the regularization parameter C of the SVR and the best results were given by **C = 1**.

Results on the validation set:

Mean Absolute Error:	0.6651754423442366
Mean Squared Error:	0.7930176131184464

The second approach – neural network with embedding layer

The first layer of the network is an Embedding layer, and so the tweets must be represented in the following manner:

- Firstly, the tweets are preprocessed as described above.
- Build the vocabulary of the tweets from the training set only (I have obtained a vocabulary containing 97 354 different words). For this, I used a dictionary to store the index of each word (the index of the first word being 1).
- The maximum length (number of words) of a tweet must be considered for the feature extraction process (in this case, the longest tweet from the training dataset is 100).
- A preprocessed tweet is represented as a list of words; the features will be obtained by replacing each word with its index from the vocabulary (in case it does not exist in the vocabulary, the word will be replaced with 0).

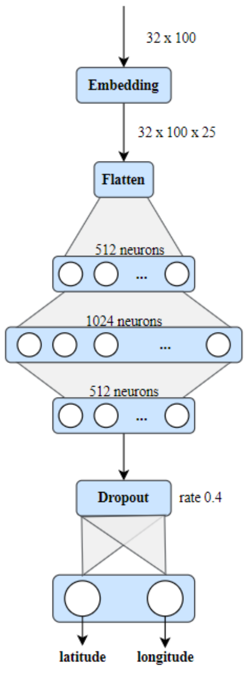
[1] German · spaCy Models Documentation

- In addition, a padding of zeros will be added to the list until its length matches the maximum length found at the previous step (for tweets longer than that value, the words at the end are discarded). This step is applied also to the validation set and the test set.

At the end of these steps, the features for each tweet should be a list of the same length as the longest tweet from the training set and consisting of integers starting from 0 to the size of the vocabulary.

The neural network:

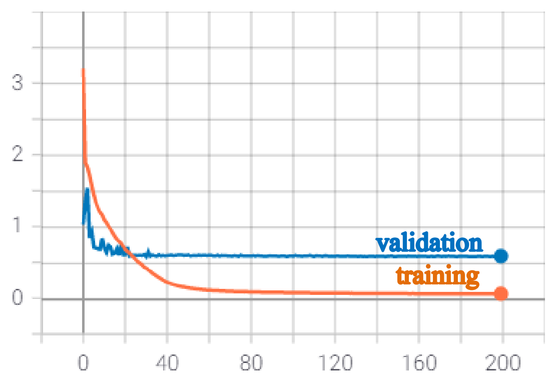
I have chosen a simple neural network using the **Sequential** model from tensorflow.keras, which contains the following layers:



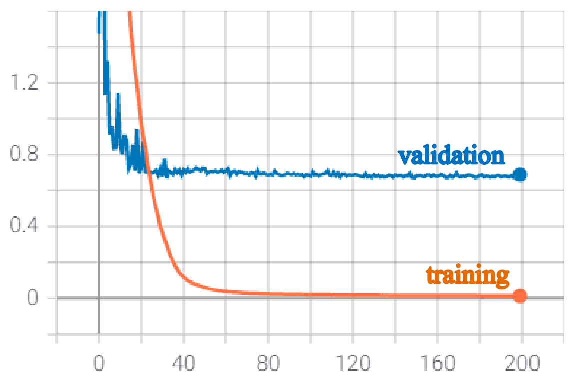
A first **Embedding** layer (I have chosen a small embedding size of 25 because larger values were not performing well) followed by a **Flatten** layer and three **Dense** layers (of sizes 512, 1024 and 512 respectively) each with **ReLu** activations; finally, I used **Dropout** of rate 0.4 as regularization method to combat over-fitting and a final Dense layer of size 2 to predict the latitude and longitude coordinates.

- The weights for the three middle Dense layers are initialized with **GlorotUniform** initializer (I also tried GlorotNormal, HeNormal and HeUniform initializers but GlorotUniform yielded the best results).
- The network is optimized using **SGD** of **learning rate 0.01** and **momentum 0.9** (I also tried different combinations of learning rate and momentum but the model's loss would not decrease as much).
- The data is split into **batches of size 32**.

I have run the model for 200 epochs and these are the results I got:



Validation Mean Absolute Error: 0.6



Validation Mean Squared Error: 0.68