UNIVERSITY OF BUCHAREST
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

PROJECT 1 - DOCUMENTATION

Geo-Location of German Tweets

Coordinating Teacher:
Professor Radu IONESCU

Student:
Raluca-Andreea GÎNGA

January, 2021
Bucharest

Table of Contents

Abstract

In this documentation, I will introduce my own methods in order to solve the task associated with the Geo-Location of German Tweets competition launched on Kaggle. This competition consisted in predicting latitude and longitude considering a wide range of texts written in German language. I formulated the purpose of the competition as a double regression problem, approaching a multitude of methods in order to minimize the mean absolute error. I applied different preprocessing of texts (manual data preprocessing, transformers for text encoding), followed by the use of the processed data in various Machine Learning models, respectively Deep Learning models. The best results were obtained with the application of a Deep Learning model with 2 Dense layers on texts encoded using Tensorflow Hub, but also with the application of a Light Gradient Boosting Machine over slightly preprocessed data. Also, Bayesian Optimization for Hyperparameter Optimization was used in order to minimize as much as possible the error.

1 Introduction

The Geo-Location of German Tweets competition had the purpose of training regression models on a dataset that was containing German tweets in order to predict geo-location. For each tweet, we had to predict its geographic location in terms of latitude and longitude. The scores were based on the mean absolute error on a given test set.

This documentation presents an approach that predict the location of a user's tweet using both Neural Networks and Machine Learning as well, both of these models being trained solely on the tweets' text content. This documentation has the intention to get good results without so many computational-intense resources.

The remainder of this documentation is organized as follows: section 2 describes the dataset used. Section 3 describes the details of the approaches used for geographical location prediction. The experiments and results on the validation set are shown in Section 4, along with Bayesian Optimization used for optimizing the results. Finally, Section 5 concludes the documentation with some conclusions about the methods proposed and the last section (6) comes with some "next" steps that could be done in order to explore more this problem and to possibly obtain better results.

2 Dataset

The Geo-Location Prediction of German Tweets competition from Kaggle provided training, validation and test data. The following table offers an overview of the data used in building the models.

	Training	Validation	Test
Number of instances	22583	3044	3138

Table 1: Overview of data

Each line of training and validation data represents a tweet and its associated coordinates separated by commas. They also have the following columns:

- the first column shows the ID of the training/validation sample.

- the second column represents the latitude of the training/validation sample.
- the third column represents the longitude of the training/validation sample.
- the fourth column is the text (tweet).

The test dataset contains only the ID of the test samples and the assigned tweet (text).

3 Methods

3.1 Light Gradient Boosting Machine

Light Gradient Boosting Machine (LightGBM) represents an efficient gradient boosting framework that is mainly based on decision-tree learning. It's an algorithm that has a faster training speed, high efficiency and a lower memory usage as well.

The mechanism of working is based on splitting the tree leaf wise with the best fit. The main difference between Light GBM and other boosting algorithms is that the latter algorithms apply the technique of splitting the tree depth wise (level wise), whereas Light GBM is leaf-wise based, that can reduce more loss than the level-wise splitting.

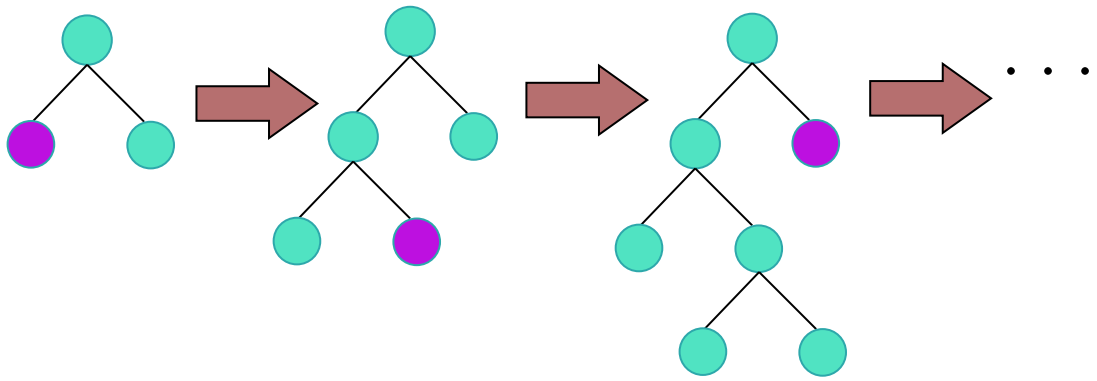


Figure 1: Leaf-wise tree growth - Light GBM

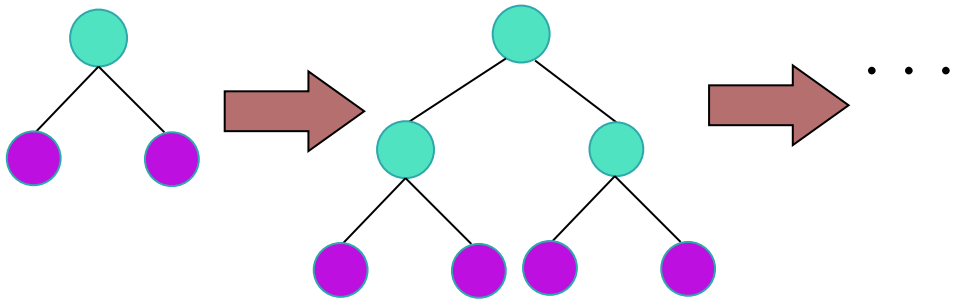


Figure 2: Level-wise tree growth - Other Gradient Algorithms (such as XGBoost)

This algorithm was used in our problem due to its fast training speed (given the fact that our data had over 20000 samples and it was very computationally expensive to apply other methods).

As a preprocessing step, I created a function that kept only the characters in the tweets (eliminating thus the emoticons and other special characters). With the obtained cleaned data, I applied a simple Count Vectorizer that provided the tokenization of this whole collection of documents and the encoding of the new documents using the vocabulary obtained. I set a maximum

of 8000 features. I noticed that putting a larger number (> 15000) was crashing the notebook, so I sticked to setting 8000 as parameter.

For predicting the latitude and longitude, I applied two Light GBM Regressors with 500 as number of estimators and a learning rate equal to 0.03, each for separately predicting the latitude and longitude.

3.2 Dense Neural Networks

Dense Neural Networks represent a type of Neural Networks where layers are fully connected (dense) by the neuron in a network layer. In this type of network, each node (neuron) in a layer receives some sort of information from all the neurons that are present in the previous layer.

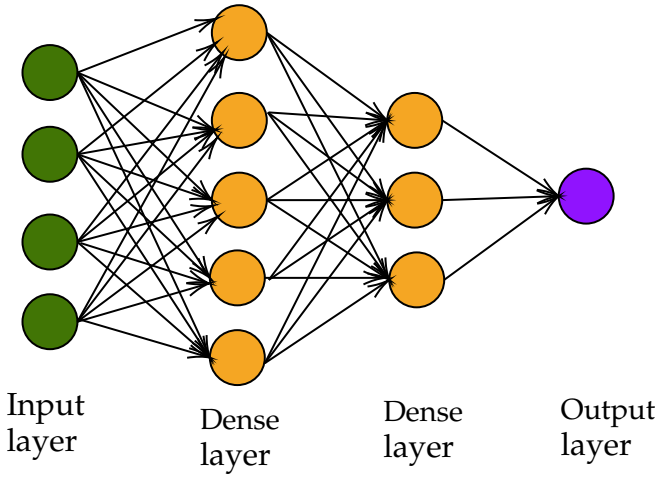


Figure 3: Fully-Connected (Dense) Layers

For building this fully-connected layer neural network, I used TensorFlow Hub [1], a repository that contains a lot of trained machine learning models that can be fine-tuned. Using TensorFlow Hub, I experimented with some encoders for text embedding.

The only text embedding that seemed to work best for our problem is Universal Sentence Encoder [2], that aims to encode the text into vectors, thus making them perfect for this problem. This model is trained and optimized for sentences, phrases and even short paragraphs, so this kind of encoder is dealing with word sequences and not with just individual words. No preprocessing was made before applying this module.

For Neural Network model, the following architecture was used: a KerasLayer where Universal Sentence Encoder was loaded from TF Hub into a Keras model, two Dense layers (one with 128 units and the other one with 64 units), with Relu activation for both of these layers (for predicting latitude) and with Leaky Relu activation function for one layer and Relu for the second one (for predicting longitude), and finally, one Dense layer for output. The optimizer was set to Adam.

3.3 Clustering

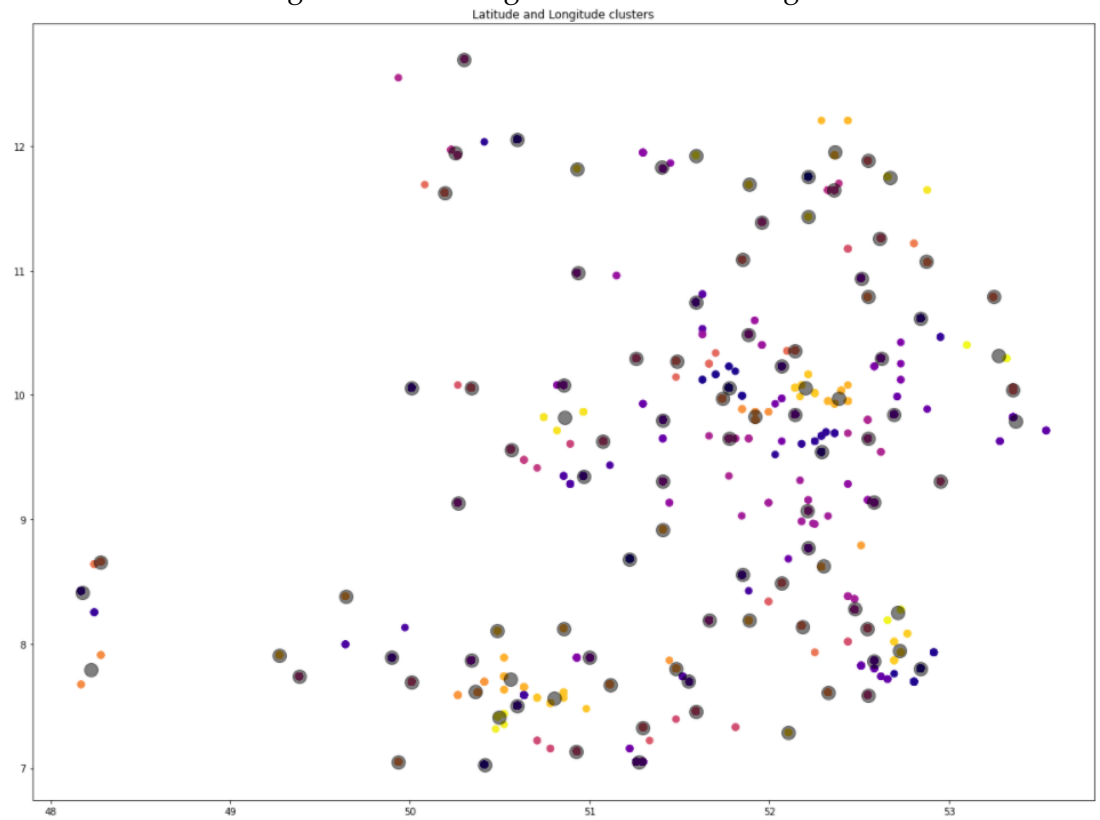
Another approach I thought about was Clustering. Initially, the purpose was to transform the regression problem into a classification one, but the computational cost was too high and the training was slow as well.

[1] https://www.tensorflow.org/hub
[2] https://tfhub.dev/google/collections/universal-sentence-encoder/1

3.3.1 Clustering of Latitude and Longitude

A first approach of clustering is the clustering of latitude and longitude and the assignment of labels for them.

Figure 4: Clustering of Latitude and Longitude



Then, for text preprocessing, I used another method by which I deleted the emoticons, special characters, punctuation marks, I turned all the words into lowercase, I deleted stopwords using an extended list of German stopwords [3] with a few additions of possible stopwords words that appeared in our texts and that had a very high frequency, but also the elimination of words that contain less than 4 letters. Stemming with Snowball Stemmer [4] was also applied. These transformations were possible with the help of the nltk [5] library specially designed for the preprocessing of textual data.

After the preprocessing part, I tried to get some insights about the popularity of the words per each cluster formed and the popularity of words in each test text as well. Then, I tried to get the number of occurences of each word from the test dataset in every cluster (that was containing popular words as well). The label with the most highest count of occurences was assigned to the label for the test dataset.

In order to transform the words in vectors, I used Tfidf Vectorizer, another method of converting a collection of documents into a matrix of TF-IDF features. Count Vectorizer was used as well for making a comparison with TF-IDF. I used 8000 features as maximum number and then proceeded with a LGBM Regressor.

[3] https://github.com/solariz/german_stopwords
[4] https://www.nltk.org/_modules/nltk/stem/snowball.html
[5] https://www.nltk.org/

4 Experiments

The results obtained from the methods proposed in the section above are in the following table:

Model	Validation	
	MAE	MSE
Light GBM	0.5707	0.5982
Fully-Connected NNs	0.4864	0.4436
Clustering	0.1046	0.0271

Table 2: Preliminary Results

Although clustering seems to give the best performance, we must keep in mind that we used two features: text tokenization and latitude and longitude formed clusters. The purpose of this method was to investigate the popularity of words per cluster and the assignment of clusters to each test sample. However, the method turned out to be a grotesque one, since it did not have good results (a MAE of 0.96642 on the test set), so I decided to eliminate this method from the proper ones. However, more research is needed to discover other similarities and to manage these clusters, but these aspects will be discussed in the conclusion, where other directions are given to further explore the issue. Thus, we decided to move on with Light GBM and with Neural Networks approaches.

4.1 Bayesian Optimization

Hyperparameters present the basis of machine learning algorithms since they control in a direct fashion the behaviors of training algorithms, having a significant effect on the performance of machine learning or deep learning models. Several techniques have been developed and successfully applied for certain application domains, such as Grid Search, Random Search, Bayesian Optimization.

We will turn our attention to a technique not very often applied in hyperparameter optimization, Bayesian optimization.

Bayesian optimization represents an efficient method of solving a series of functions that are computationally intense to find global extremes. In our problem, the goal of optimization is to find those parameters $\hat{x}$ that minimize a function $f(x)$ over a domain A

$$\hat{x} = \arg\min_{x \in A} f(x) = \arg\max_{x \in A} (-f(x)) \ ,$$

where A is the search space of x.

Bayesian optimization has two main directions: it incorporates previous data about function f and updates previous information with samples extracted from f to obtain a posterior that is closer to objective function F. We call surrogate model that model that can be a good approximation of the objective function. Bayesian optimization has the purpose to use an acquisition function that can easily drive to the area where it is likely to happen an improvement than the present observation. At each step, Bayesian Optimization determines what is the best point to evaluate next in concordance with the acquisition function by optimizing it. Then, the model is updated and the process continues to determine the next point that needs to be evaluated.

Our problem was to find those parameters that minimize the mean absolute error function, that means maximizing the negative of mean absolute error.

Thus, I used the Bayesian Optimization package [6] and tuned the promising models: LightGBM on Machine Learning and the Neural Network models with 2 Dense layers.

[6] https://github.com/fmfn/BayesianOptimization

For Light GBM, our Bayesian Optimization process gave the following parameters in order to obtain the best results for Latitude and Longitude predictions. We should note that we trained two separate regression problems, one for Latitude prediction and the other one for Longitude prediction.

We obtained the following parameters:

	Latitude							
	bagging_fraction	feature_fraction	learning_rate	max_depth	min_child_weight	min_split_gain	num_iterations	num_leaves
Light GBM	0.9	0.5	0.03	-1	5	0.03	2000	500

Table 3: Parameters given by Hyperparameter Optimization for Latitude prediction

	Longitude							
	bagging_fraction	feature_fraction	learning_rate	max_depth	min_child_weight	min_split_gain	num_iterations	num_leaves
Light GBM	0.8	0.4	0.01	-1	20	0.05	3000	300

Table 4: Parameters given by Hyperparameter Optimization for Longitude prediction

To find the best parameters of the neural network, I used Bayesian Optimization again to find the most appropriate values for the nodes in the two Dense layers, but also to find the alpha value in Leaky Relu (used to predict longitude).

I found out the following parameters:

$$\begin{aligned} Latitude &: \text{122 for first Dense Layer, 32 for second Dense Layer} \\ Longitude &: \text{422 for first Dense Layer, 248 for second Layer and 0.6923 for alpha} \end{aligned}$$

The scores after Hyperparameter Optimization (HPO) are listed in the following table:

Model	Validation	
	MAE	MSE
Light GBM	0.5576	0.5742
Fully-Connected NNs	0.5056	0.5706

Table 5: Models scores after HPO

5 Conclusion

In the current project, we solved the problem posed by the Geo-Location of German Tweets competition on Kaggle [7]. I applied various methods, including the application of a Light Gradient Boosting Machine over superficial preprocessed data, the application of a neural network model with the transformation of given texts into embeddings using TensorFlow Hub, a library not so well known and often applied in Deep Learning problems, but also the approach of a method of clustering the values we had to predict. The enormous potential was given by Light GBM and the neural network model, so I tried tuning the models with Bayesian optimization. The best result on mean absolute error was provided by the neural network model, on the test data on Kaggle obtaining a score of 0.51117 on the public leaderbord and 0.51270 on the private leaderbord, so not a significant jump that ensures the generality of the model.

6 Next Steps

As a future work for the detailed exploration of the problem, I propose the following solutions (with the mention that some approaches have been tried by me, but due to GPU and not having

[7] https://www.kaggle.com/c/pml-2020-unibuc/

a larger RAM, many processes have failed both on the personal laptop and on Google Colab):

- Transforming the regression problem into a classification one clustering the latitude and longitude. From previous research, I saw a number of clusters equal to 150 is good for our problem (using both silhouette scores and grouping the latitude and longitude as well). [1]
- Trying $\nu - SVR$ based on string kernel, as proposed by Găman and Ionescu [2]
- Exploring more Character-Level Convolutional Neural Network (CNN) and see if this method works [2]
- Creating an embedding matrix using FastText, Doc2Vec or Word2Vec for further use in Deep Learning models or even Machine Learning algorithms
- Trying multiple feature extraction approaches (word, char and char_wb as analyzers) with various number of features and n-gram range

Bibliography

[1] Fernando Benites, Manuela Hurlimann, Pius von Daniken, Mark Cieliebak, *ZHAT-InIT - Social Media Geolocation at VarDial 2020*, International Committee on Computational Linguistics (ICCL), 2020

[2] Mihaela Găman, Radu Tudor Ionescu, *Combining Deep Learning and String Kernels for the Localization of Swiss German Tweets*, arXiv:2010.03614v1, 2020

[3] Peter I. Frazier, *A tutorial on Bayesian Optimization*, arXiv:1807.02811, 2018.

[4] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, Si-Hao Deng, *Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization*, Journal of Electronic Science and Technology, New Jersey, 2019.