# Sequence and Anger Regression using Transformers, Contextualized and Transformed

**Abstract**
This report examines two tasks. On the one hand we performed a named entity recognition task using different models derived from the BERT-base model. We created 3 fine-tuned versions of the system, of which the model Fine-tuned with 3,000 sentences outperformed the others. Further we determined that freezing has no significant impact on the performance of the model. Therefor not freezing is the better choice. On the other hand we performed a regression analysis, using different models to determine the emotional state of an individual. The roBERTa-based

**Keywords**
Named Entity Recognition — BERT — Regression

## Contents



## 1. Introduction

This is our report for the fifth assignment of the Machine Learning for Natural Language Processing 1 course. This exercise introduces you to named entity recognition and transformer-based neural architectures. The goals of this exercise are to:

- understand NER as a task and the IOB-format
- and to be able to use pretrained models like BERT and fine-tune them on a specific task (using HuggingFace, Pytorch, TensorFlow).

## 2. Data & Task Description

This section discusses the data we worked with and the task we completed. First the data is discussed and then the task is elaborated.

### 2.1 Data

The task description states that for this exercise, you will work with a small part of the polyglot-ner dataset, described in this paper; and with a documentation of the corresponding python-package. However, these resources are just for reference, you can import the dataset into Python directly using the 'datasets'-library.

- Link to Paper: https://arxiv.org/abs/1410.3791
- Link to Documentation: https://polyglot.readthedocs.io/en/latest/NamedEntityRecognition.html
- Datasets Library: https://huggingface.co/docs/datasets/quickstart

The URLs above provide all important data and background information for this exercise. Further the task states that the polyglot-ner dataset contains 40 languages. Choose one language to work with from of that dataset. The following conditions need to hold for the language:

- It is not English.
- There must exist a pretrained Hugging Face BERT-base model for the language (an alternative solution is a multilingual model).
- The polyglot-ner dataset needs to contain at least 7000 sentences in this language.

Using the datasets-library to extract two training sets, one containing 1,000 sentences and one containing 3,000 sentences. Also, extract an evaluation set (in this exercise, we don't differentiate between development and test set) of 2,000 sentences.

### 2.2 Named Entity Recognition using BERT

Implement a named entity recognition system for your chosen language. Use HuggingFace's BertForTokenClassification-class and initialize it with a pretrained Hugging Face BERT-base model of your chosen language. This HuggingFace guide for fine-tuning serves as a good starting point. Before passing the data to the model, you need to encode it using a HuggingFace tokenizer. Use the tokenizer corresponding to your BERT model. When provided with the right arguments, the tokenizer can also pad and truncate the input. You can reduce the amount of code for this exercise by using the Trainer class explained at the bottom of the HuggingFace guide. You will create 3 fine-tuned versions of the system:

1. Fine-tuned with 1,000 sentences
2. Fine-tuned with 3,000 sentences
3. Fine-tuned with 3,000 sentences and frozen embeddings.

Let each fine-tuned model predict on the evaluation set to compute f1-micro and f1-macro scores. Then, answer:

1. When initializing the BertForTokenClassification-class with BERT-base you should get a warning message. Explain why you get this message.
2. Which model performed best on the evaluation set?
3. Are there differences between f1-micro and f1-macro score? If so, why?
4. How large is the performance gap between 1'000 and 3'000 sentences for finetuning?
5. Is it better to freeze or not to freeze the embeddings?

### 2.3 Part 2: Regression Competition: How angry are you?

BERT is the eldest Transformer and it is still very popular and very well-performing. But, of course, there are many more transformers out there and some seem to be performing better on different tasks. In the last exercise, you used CNN's to explore the emotion detection task. Now that you have the data loading ready, we ask you to adapt it to read the Regression set of the same dataset in English. The idea of this part is straightforward. We expect you to submit a fine-tuned Transformer model for the Affect of Anger in the English Regression task. We have uploaded the ".txt files" of the dataset in the exercise's material folder. We hope you will explore a few models (at least briefly read through their papers and choose one to train). Of course, you can also do this experimentally. We leave this decision to you. You are allowed to complete Part 2 by using simpletransformers (easier to implement) by following their Regression Guide. Of course, you can always use HuggingFace just like you did for Part 1.

Tip: Preprocessing also matters when adapting to the Twitter domain ;)

Please have a statement in your report as to why and what hypothesis led you to choose this architecture. Defend why you chose/chose not to do any preprocessing? Did your results support the hypothesis? Why/Why not?

## 3. Data Preprocessing

In this section we discuss the different measures that we took to preprocess the data. First we discuss the NER task and then the regression task.

### 3.1 NER

We decide to use "German" for the name entity recognition. This choice was made based on the plenitude of samples (547578 in total). The dataset can be loaded using the function *dataset.load* in huggingface. The dataset has a dictionary-like structure. Each sample, contains four features:

- **'id':** The index of each sample in the dataset.
- **'lang':** The language of each sample.
- **'words':** The words in the sample, which are structured as a list.
- **'ner':** The name entity for each word, which is structured as a list. The types of entities are 'ORG' for the name of organizations, 'LOC' for the name of locations, 'PER' for the name of people, and '0' for others.

As part of preprocessing, we take the following steps:

- **Tokenization:** We use the pretrained tokenizer from "bert-base-german-cased" to transfer the original text words to their ids.
- **Covert labels:** As the string type labels aren't supported by $sklearn\hat{f}1\_score$, we need to convert them to integer-type indexes.

### 3.2 Regression

Similar to the last exercise, the training and testing datasets are located in different ".txt files" with different filenames. The training dataset contains 1701 rows while the testing dataset contains 1002 rows. We use the column *Tweet* as the input text. The following steps are taken to preprocess the texts in both datasets:

1. **Remove all HTML tags.**
2. **Convert all characters to lowercase.**
3. **All emojis were converted to their textual meaning.** This decision was made because emojis reflect what an writer is feeling in a moment. While they are irrelevant for tasks like language recognition in the context of emotion recognition they play a very important role.
4. **Remove all punctuation.**

The module Simple Transformers helps us tokenize the text automatically, so there's no need to tokenize the text manually. We use the column *Intensity Score* as labels for the task, so there is no need to further process it. Figures 1 and 2 show the distribution of label values in the two datasets we just described.

**Figure 1.** Distribution of labels in training dataset

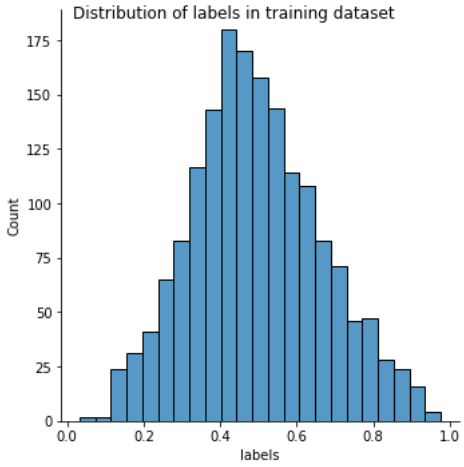
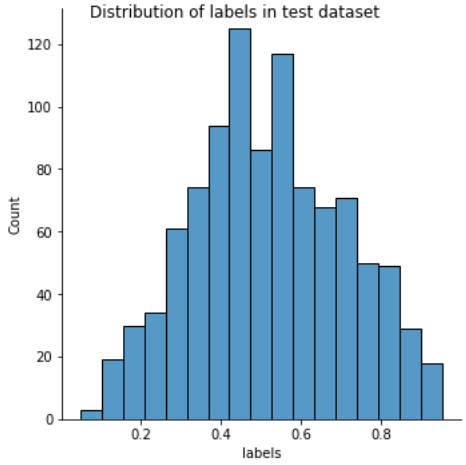



**Figure 2.** Distribution of labels in test dataset




## 4. Models

In this section we discuss the different models used in this exercise. The task makes use of both NER and regression. We will first discuss NER and then explain the regression.

### 4.1 NER

For the task at hand, we decided to import the "bert-base-german-cased" model. This bert-based model is trained, using the *Wiki, OpenLegalData, News (about 12GB)* dataset.

### 4.2 Regression

We analyzed four different models, namely Bert-base, RoBERTa-base ,DistilBERT base uncased finetuned SST-2 and Twitter-roBERTa-base for Emotion Recognition

- **BERT-base** is a Transformer model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on raw texts only, with no human interfering in the labeling process in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. We choose it as a base model to compare the performances of different models.
- **RoBERTa-base** This model improves upon the BERT model in the following four aspects: (1) the model is training over a longer period of time, with bigger batches, using more data; (2) the next sentence prediction objective has been removed; (3) it has been trained on longer sequences; and (4) the masking pattern applied to the training data is dynamically changed. Researchers also collected a large new dataset (CC-NEWS) of comparable size to other privately used datasets, to better control for training set size effects. We hypothesize that this model will have a better performance than BERT.
- **DistilBERT base uncased finetuned SST-2** is based on the DistilBERT base model, which is the distilled version of the BERT base model and is later fine-tunned using the Stanford Sentiment Treebank (SST). The SST consists of sentences from movie reviews and human annotations of their sentiment. According to this paper, compared to the original BERT model, the distilled version pretrains a smaller general-purpose language model and is able to reduce the size of a BERT model by 40%. As the regression task is similar to a 2-class classification, we thought this lightweight and efficient version could fit our task well and improve the performance.
- **Twitter-roBERTa-base for Emotion Recognition** is a roBERTa-base model trained on 58M tweets and fine-tuned for emotion recognition using the TweetEval benchmark. We chose it because it has been specifically designed for emotion recognition and should be able to perform well in this task.

For the first three models in our list, we argue that it is necessary to do data preprocessing. We have come to this conclusion because they were not originally designed for emotion classification, and therefore certain symbols or emojis may not be processed in the correct fashion. To avoid this, we preprocess the data as described in the previous chapter. For the last model in our list, using the original tweets should work fine because it has already been fine-tuned to fit emotion classification. Having read up on this model, the documentation shows that this model can identify symbols and emojis.

## 5. Model Training and Evaluation

This section is composed the same as the previous section. First we report on the NRE and then on the regression.

### 5.1 NER

For all our models we used the same hyper-parameters. Table 1 illustrates the combination while Table 2 shows the F1 macro and micro scores and our model accuracy. We kept the number

| Hyper-parameter | Value |
|---|---|
| learning rate | 1e-4 |
| weight decay | 1e-5 |
| training batch size | 10 |
| testing batch size | 10 |
| epochs | 2 |

**Table 1.** Hyper-parameter combination used for the models.

| Model | F1 Macro | F1 Micro | Accuracy |
|---|---|---|---|
| Fine-tuned 1,000 | 0.690491 | 0.940616 | 0.940616 |
| Fine-tuned 3,000 | 0.715436 | 0.942351 | 0.942351 |
| Fine-tuned 3,000 frozen Emb. | 0.712426 | 0.943464 | 0.943464 |

**Table 2.** F1 micro and macro scores and accuracy.

of epochs low in order to decrease training time. The model labeled "Fine-tuned 3,000 frozen Emb." performed the best.

### 5.2 Regression

The four models use the same combination of hyper-parameters during training, which is illustrated in Table 3. This resulted in the scores shown in Table 4. Here we trained the fourth model for 20 epochs, which we assume lead to it achieving a higher score. The Pearson r- score is 0.826 and r2 score is 0.680.

| Hyper-parameter | Value |
|---|---|
| learning rate | 4e-5 |
| weight decay | 1e-4 |
| train batch size | 64 |
| epochs | 10 |
| early stopping | Yes |

**Table 3.** Hyper-parameter used in model training.

| Model | Pearson-r score | r2-score | eval_loss |
|---|---|---|---|
| BERT-base | 0.768 | 0.588 | 0.0147 |
| RoBERTa-base | 0.802 | 0.632 | 0.0131 |
| DistilBER SST-2 | 0.737 | 0.533 | 0.0167 |
| roBERTa Emo. | 0.815 | 0.664 | 0.0120 |

**Table 4.** R-scores and eval_loss of the different models.

## 6. Discussion

As the previous chapters this chapter also follows the same setup. We first discuss the NRE task and then the regression.

### 6.1 NER

1. **Warning when initializing** When initializing the BertForToken Classification-class with BERT-base we get the following warning message (we decomposed the list items to visualize them in an easy-to-read manner):
   - Some weights of the model checkpoint at bert-base-german-cased were not used when initializing BertForTokenClassification: ['cls.predictions.transform.LayerNorm.weight' ,'cls.predictions.transform.dense.bias' , 'cls.predictions.bias' , 'cls.predictions.transform.LayerNorm.bias' , 'cls.seq_relationship.bias' , 'cls.seq_relationship.weight', 'cls.predictions.transform.dense.weight', 'cls.predictions.decoder.weight']
   - -This is to be expected if you are initializing BertForTokenClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
   - -This is not to be expected if you are initializing BertForTokenClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
   - Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-german-cased and are newly initialized: ['classifier.bias', 'classifier.weight']
   - You should probably train this model on a downstream task to be able to use it for predictions and inference.

   We think that we receive this error message because the BERT model 'bert-base-german-cased' (that we want to use for our BertForTokenClassification model) has been pre-trained on another architecture (BertForPreTraining). This means that the layers of our target architecture (BertForTokenClassification), which are not present in our base architecture (BertForPreTraining) will be randomly initialized. The layers of our base model (BertForPreTraining), which are not part of our target architecture (BertForTokenClassification) will be discarded. This means that the model has to be fine-tuned on a downstream task in order to learn the proper weights in the randomly initialized layers.

2. **Best performance** We think the **Fine-tuned with 3,000 sentences without frozen** outperformed the rest. We come to this conclusion due to it having the highest f1-macro score and a similar f1-micro score as Fine-tuned with 3,000 sentences and frozen embeddings.

3. **Difference between f1 micro and macro** There are big differences between micro and macro f1. The macro f1 metric is computed by independently computing the f1 score for each class/ label separately and then taking the average (In this case all classes are weighted equally).

The micro f1 is computed by weighting the f1 score of every class by the number of samples supporting this class (In this case the weight of a class is proportional to its number of samples). The big negative deviation between micro and macro f1 just says that the model is performing well for frequent classes of labels while performing worse for at least one minority class.

4. **Performance gap between model trained by 1000 and 3000** There is a performance gap between a BERT model being fine-tuned on 1000 and 3000 sentences of approx 0.2% micro f1 and approx 2.5% macro f1.

5. **To freeze or not to freeze** Actually our results show that there's no significant difference between the performance of the two models. As the non-frozen model gets 0.1% lower on micro f1 but 0.3% higher on macro f1, we think it's better not to freeze the embedding layers.

### 6.2 Regression

After training, we noticed that the performance of RoBERTa is better than BERT, which proves that the improvements made in RoBERTa also improve the performence and was to be expected. DistilBERT performs worse than BERT. This could be attributed to its lightweight, which may lead to underfitting. Further it is not originally designed for emotion classification, meaning it can be expected to underperform in this task compared to models designed specifically for this task. The Twitter-roBERTa-base performs best. This is not surprising and can be attributed to its fine-tuning for emotion classification, which fits the task best. One of the reasons this model outperforms the others may be that it potentially uses further preprocessing components tailored to this task.

## 7. Conclusion

In conclusion, as expected, the Twitter-roBERTa-base outperformed the other models. We determined that freezing does not make a significant difference regarding the performance and therefore not freezing is the way too go.