Machine Learning for NLP 1

Exercise 5

Sequence and Anger Regression using Transformers

University of Zurich

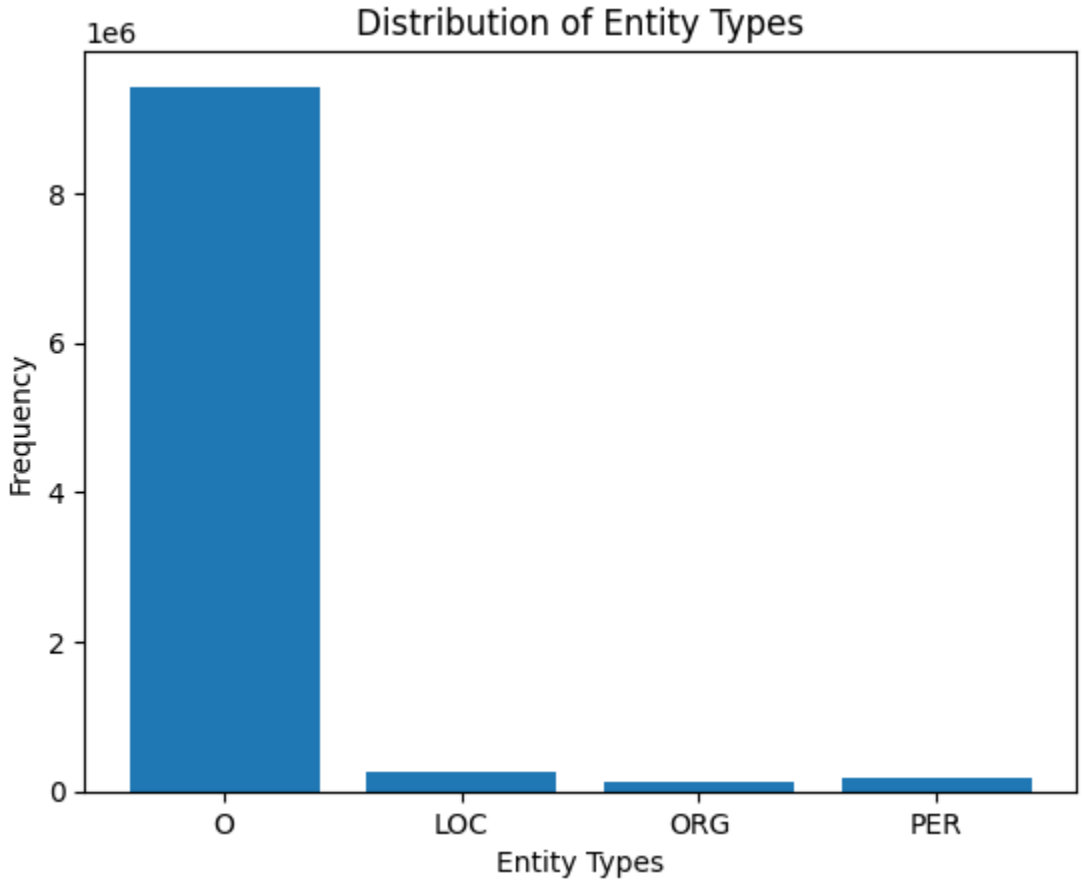
Contents

Problem Statement

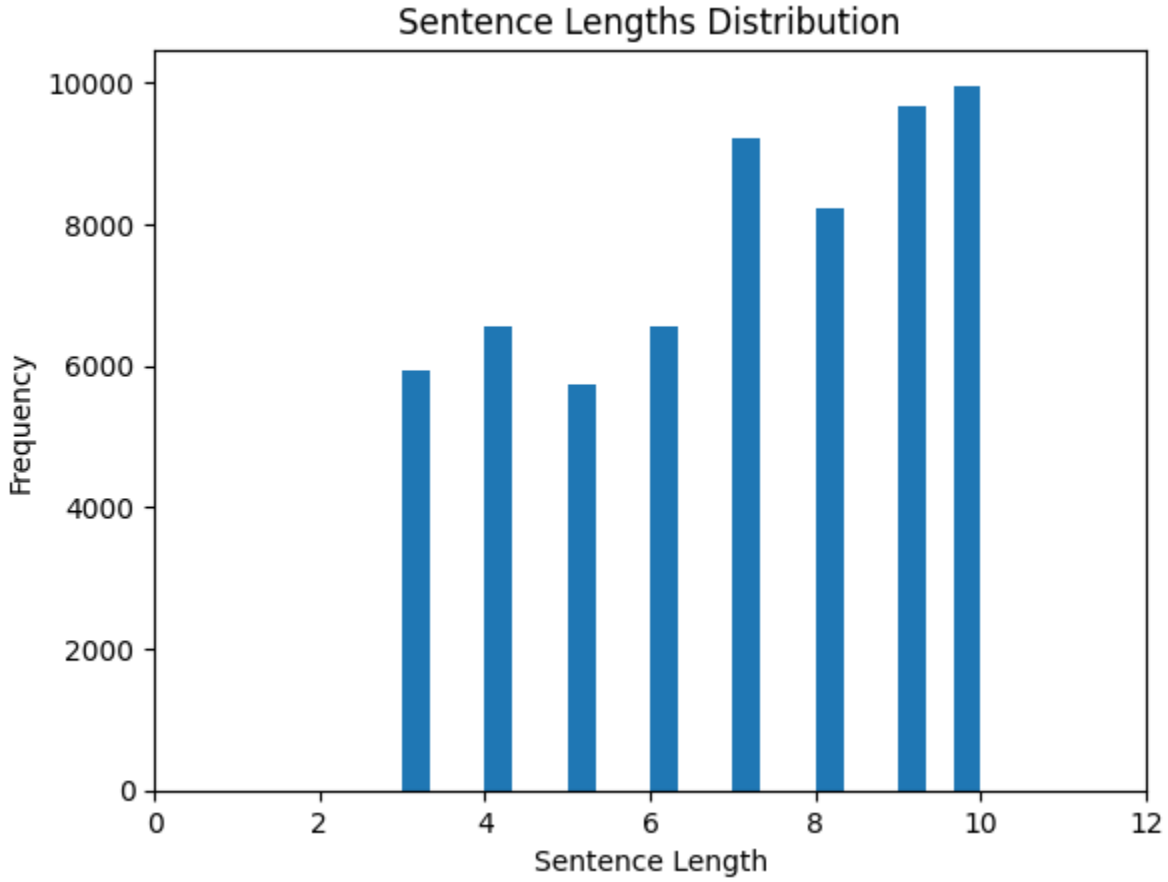
In this exercise we apply transformer-based neural architectures to two distinct tasks: Named Entity Recognition (NER) and Emotion Regression. In the NER part, we implemented and fine-tuned a system using Hugging Face's BertForTokenClassification class for French language exploring the effects of training dataset size and embedding freezing on model performance. The second part focuses on emotion detection, where we fine-tuned an alternative transformer model to assess the emotion of anger in English texts.

Exploratory Data Analysis (EDA)

We perform a brief Exploratory Data Analysis (EDA) on the train dataset. The train dataset has 418411 samples and the distribution of Entity Types are as follows.



We also plot the frequency of samples with regards to their sentence lengths.



PART 1

Vectorizer and Model

For this exercise we will be using the tokenizer and the pretrained model from *bert-base-multilingual-cased*. This model supports the French language for NER token classification which solves our purpose.

We use 3 epochs to train our models with 500 warm up steps and a weight decay of 0.01.

In our final model, we will freeze the embeddings module of our model using *param.requires_grad = False* for the *model.bert.embeddings.parameters()*.

```
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(119547, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=4, bias=True)
)
```

Results and Scores

Model with 1000 Training Samples

F1-Micro Score: 0.9564681724845996

F1-Macro Score: 0.6351189098363828

Model with 3000 Training Samples (Unfrozen Embeddings)

F1-Micro Score: 0.9643037426984928

F1-Macro Score: 0.7414053035339611

Model with 1000 Training Samples (Frozen Embeddings)

F1-Micro Score: 0.9617048436281115

F1-Macro Score: 0.7513784810993189

Discussion

Q.1 Warning Message When Initializing BertForTokenClassification with BERT-base: This warning occurs because the BERT-base models are pre-trained for a general language understanding task, not specifically for NER. When initializing BertForTokenClassification, it adapts this pre-trained model for token classification, which involves modifying the model's head to suit the NER task. This modification triggers a warning since the pre-trained weights were not originally trained for token classification.

Q.2 Best Performing Model on the Evaluation Set: The best performance is seen in the model fine-tuned with 3,000 sentences and unfrozen embeddings, with an f1-micro score of 0.9643 and f1-macro score of 0.7414. While the model with frozen embeddings also shows strong performance (f1-micro: 0.9617, f1-macro: 0.7514), it slightly lags behind the model with unfrozen embeddings.

Q.3 Differences Between f1-micro and f1-macro Score: The difference between these scores is due to their calculation methods. f1-micro calculates the metric globally across all classes, making it more influenced by the model's performance on the most frequent labels. f1-macro, on the other hand, computes the metric independently for each class and then takes the average, which means it treats all classes equally, regardless of their frequency in the dataset.

Q.4 Performance Gap Between 1,000 and 3,000 Sentences for Fine-Tuning: Comparing the models trained with 1,000 and 3,000 sentences, the model trained with 3,000 sentences (unfrozen embeddings) shows a noticeable improvement, indicating that a larger training dataset contributes to better model performance. The increase in both f1-micro and f1-macro scores suggests enhanced overall model accuracy and a more balanced performance across different entity types.

Q.5 To Freeze or Not to Freeze the Embeddings: Based on the results, it appears slightly better not to freeze the embeddings. The model with unfrozen embeddings achieved the highest scores (f1-micro: 0.9643, f1-macro: 0.7414), although the difference is not substantial compared to the model with frozen embeddings (f1-micro: 0.9617, f1-macro: 0.7514). This suggests that allowing the embeddings to update during training can provide a marginal improvement, possibly by better adapting to the specific characteristics of the NER task. However, the relatively high performance of the model with frozen embeddings also indicates that the pre-trained embeddings already capture useful information for the task.

PART 2

Architecture Choice and Hypothesis

The decision to use the BERTweet model, specifically the "vinai/bertweet-base" variant, for the emotion regression task was driven by a hypothesis centered around the nature of the dataset.

Given that the dataset comprises Twitter data, a platform characterized by informal language, abbreviations, and unique lexical items like hashtags and mentions, a model specialized in understanding the nuances of Twitter's language structure seemed ideal. BERTweet is pre-trained on a large corpus of Twitter data, making it inherently more adept at capturing the linguistic idiosyncrasies of social media text compared to standard BERT models.

Results and Hypothesis Validation

The results seem to support this hypothesis. The Pearson correlation coefficient of 0.7985 on the test set is a robust indicator of the model's performance. This metric measures the linear correlation between the predicted and actual values, with a score closer to 1 indicating a strong positive correlation. In this context, a score of approximately 0.8 suggests that the model predictions align well with the actual data, indicating effective understanding and processing of the Twitter text. This outcome validates the initial hypothesis that a Twitter-specific model like BERTweet, with minimal preprocessing, would be well-suited for accurately capturing and predicting emotional nuances in Twitter data.