Assignment 2 - Natural Language Processing

Summary of the assignment: Transformers, Regression and Quantization
Author: Maruan Bakri Ottoni, maruan.ottoni@ime.usp.br
Número USP: 13710368

1 Abstract

BERT models are important models for natural language processing and widely used on the academia and industry for specific tasks by fine tunning it. On this work we focused on trying to understand how to fine tune the BERTimabu model, a variant of BERT for Portuguese, for two specific tasks: regression of density of vowels in a text and classification of sentences. We achieved good performances measured by different metrics on the two tasks. The code can be runned here Colab.

2 Introduction

The BERT model was proposed in BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. It's a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia. The goal of this article is to explore the refinement of encoders in the Transformers architecture (BERT-like). We will also examine the use of this architecture in the context of numerical regression. Neural networks can be used for both regression and classification, but in the Transformers (BERT-like) architecture, they are fully trained in the task of classification. That is, the pre-training tasks in predicting masked words are tasks of classifying vectors into a very large number of classes, which correspond to the words.

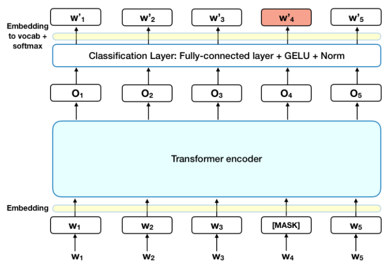
On the other hand, sentence order detection tasks are binary classifications. Therefore, it is not surprising that networks pre-trained for classification do not achieve very good results in regression tasks. The objective of this work is to draw attention to this fact and to propose possible solutions. One way to mitigate the problem of low performance in regression is quantization, that is, transforming a continuous range of values into possibly varied size bands, in order to generate a small number of categories, transforming a regression problem into a classification problem

On all the experiments in this article we used the corpus of reviews of B2W, and focused on the *review text* column for our experiments.

3 Preliminaries

Traditional NLP models often struggled with capturing the context of words in sentences, limiting their effectiveness. The advent of BERT marked a paradigm shift by introducing a mechanism that considers the context of a word in both directions (left and right of the word in a sentence), fundamentally altering how language models understand textual data.

BERT's architecture [1] is based on the Transformer model, specifically its encoder component. The Transformer, originally proposed by Vaswani et al [2], departs from previous sequence learning methods by relying entirely on attention mechanisms, discarding recurrent layers. BERT utilizes this architecture to process each word in the context of all other words in a sentence, a stark contrast to earlier models that processed words sequentially. On figure 1 is a schematic of the BERT architecture.



w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 1: A simple schematic illustrating the basic architecture of the BERTlike family of transformers.

BERT comes in several variants, notably BERT-Base and BERT-Large, differing in their layer counts, hidden units, and attention heads. BERT-Base comprises 12 transformer layers, 768 hidden units, and 12 attention heads, while BERT-Large consists of 24 transformer layers, 1,024 hidden units, and 16 attention heads.

BERT's training process consists of two stages: pre-training and fine-tuning. During pre-training, the model is trained on a large corpus of text on two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly masks a percentage of input tokens and the model learns to predict these masked tokens. NSP involves the model predicting whether a sentence logically follows another.

The pre-training phase allows BERT to develop a deep understanding of language context and relations. As it learns to predict the missing words and the relationship between sentences, the model captures both syntactic and semantic nuances.

In fine-tuning, BERT is adapted to specific NLP tasks, such as question answering, sentiment analysis, and language inference. This is achieved by adding a task-specific layer on top of the pre-trained model and training it on labeled data relevant to the task.

The hallmark of BERT is its bidirectional nature. Unlike directional models, which read the text in a fixed order (left-to-right or right-to-left), BERT reads the entire sequence of words at once. This allows for a more nuanced understanding of context, as the meaning of each word can be influenced by all other words in the sentence.

BERT has significantly advanced the state-of-the-art in various NLP tasks. It has been successfully applied in question answering systems, language translation, summarization, and named entity recognition, demonstrating remarkable improvements over previous models.

4 Vowels regression

The regression task that we will predict with Transformer neural networks will be the density of vowels in a text excerpt (DV). This numerical value is obtained by considering only the characters that correspond to the letters of the Portuguese alphabet, uppercase or lowercase [A-Z,a-z,Ç,ç,Ã,ã,...], counting the number of vowels with or without accents, and dividing by the total number of letters. In this way, we will be ignoring all space and punctuation characters, as well as digits and other non-letter characters.

For example, in the sentence taken directly from the B2W-reviews corpus: 'Meu filho amou! Parece de verdade com tantos detalhes que tem!' This sentence has 63 characters, but discounting the white spaces and punctuation marks, it has only 50 letters, of which only 23 are vowels. Thus, the vowel density of the sentence is 0.43.

It is important to note that when tokenization followed by embedding is done, the characters that make up the word are no longer present, so the vowel density is a completely abstract number, independent of the language and the style of text.

The final model is illustrated below:

```
BertForRegression(
  (bert): BertForClassification(
    (bert): BertModel(
      (embeddings): BertEmbeddings(
        (word_embeddings): Embedding(30522, 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)
            )
          )
        )
      )
      (pooler): BertPooler(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (activation): Tanh()
      )
    )
    (classifier): Linear(in_features=768, out_features=3, bias=True)
  )
  (regressor): Linear(in_features=768, out_features=1, bias=True)
)
```

Listing 1: BertForRegression model structure

4.1 Discussion

In preprocessing, we calculated the vowel density of each of the sentences in your corpus, both in the training part, as well as in the validation part, and in the test part. The loss function utilized for the regression task was the MSE (Mean squared Error). For computational constraints, we only used 10000 samples of the dataset and splitter this samples into 6000 training samples, 2000 validation samples and 2000 test samples. For the training we used the adam optimizer and a batch size of 16. We only trained the model for 3 epochs due computational constraints.

On out experiment we evaluated the results by different references and different metrics as well. We utilized the RMSE, MAE, MAPE, R2 and the Pearson correlation to evaluate the trained model on the test set. The ground truth of the predictions was varied as well. Here we utilized the vowel density per sentence, the vowel density of the first word of the sentence, the vowel density of the last word of the sentence and the vowel density of the entire train data. The results are summarized on the table 1

Method	**RMSE**	**MAE**	**MAPE**	**R2**	**Pearson**
Using only the first word	0.220737	0.144474	41496.688843	-1.597408e-01	0.028082
Using only the last word	0.155576	0.099131	324313.549805	-1.580826e-01	-0.001212
Using the global density	0.005572	0.004293	0.885921	-3.488848e+10	NaN
Using the density per sentence	0.041944	0.023667	14563.088989	-1.658915e-02	0.044086

Table 1: Summary of the methods used for regression on the fine tuned BERT model. Here we highlight different metrics and different gound truth for the vowel density on the method column.

Using only the first word:

RMSE: 0.220737 and MAE: 0.144474: These values indicate moderate errors in prediction. The RMSE value, being higher, suggests the presence of some significant errors, as RMSE gives more weight to larger errors. MAPE: 41496.688843%: An extremely high MAPE suggests that the model is highly inaccurate for this method, perhaps due to the limited context provided by a single word. R2: -0.1597408: A negative R2 value indicates that the model performs worse than a simple mean-based prediction. This could be due to the insufficient context when only the first word is considered. Pearson: 0.028082: A low Pearson coefficient implies a very weak linear relationship between the predicted and actual values.

Using only the last word:

RMSE: 0.155576 and MAE: 0.099131: These values are lower than those for the first word, suggesting slightly better accuracy. However, the errors are still considerable. MAPE: 324313.549805%: Like the first word, the extremely high MAPE here suggests that the last word provides insufficient context for accurate prediction. R2: -0.1580826: Similar to the first word, the negative R2 value indicates poor model performance. Pearson: -0.001212: This value is close to zero, showing no linear correlation between predictions and actual values.

Using the global density:

RMSE: 0.005572 and MAE: 0.004293: These are the lowest error metrics among all methods, suggesting high accuracy. MAPE: 0.885921%: This low percentage indicates very high accuracy in prediction. R2: -3.488848e+10: The extremely large negative value is unusual and might indicate an error in calculation or an anomaly in the data. Pearson: NaN: The 'NaN' value suggests that the Pearson correlation coefficient couldn't be calculated, possibly due to constant actual values in the test set. This makes sense since the global density does not vary along its values.

Using the density per sentence:

RMSE: 0.041944 and MAE: 0.023667: These errors are higher than those for the global density but much lower than those for individual words, indicating moderate accuracy. MAPE: 14563.088989%: A high MAPE, though significantly lower than for single-word methods. R2: -0.01658915: A slightly negative R2 value indicates that the model does not perform much better than a mean-based prediction. Pearson: 0.044086: A low but positive Pearson coefficient suggests a weak linear relationship.

Overall the model performs best when considering global density, suggesting that a broader context is crucial for accurate vowel density prediction. Using individual words (either the first or the last) for prediction results in significant errors and low correlation with actual values, underscoring the importance of context in NLP tasks. The negative R2 values in most methods (except global density) indicate that the model struggles to outperform a basic mean predictor. This could point to limitations in the model's training or the complexity of the task. The Pearson coefficients are generally low, suggesting that the predictions do not have a strong linear relationship with the actual values, except in the case of global density which couldn't be computed. The results highlight the challenges in vowel density regression tasks and the importance of sufficient context for models like BERT to perform effectively. This analysis should guide future modifications and improvements in the model and approach.

5 Quantization

We will perform two tasks of quantized classification, in which we will have three categories to classify the density of vowels in a given sentence. We will classify a sentence according to three classes. In Class 1 are the sentences with densities lower than 1/3; in Class 2 are the sentences with densities between 1/3 and 2/3; and in Class 3 are the sentences with densities greater than 2/3.

Note that the classes are not balanced, and it is very likely that Class 2 contains almost all of the sentences in the corpus. To illustrate this we plotted the histogram of 2

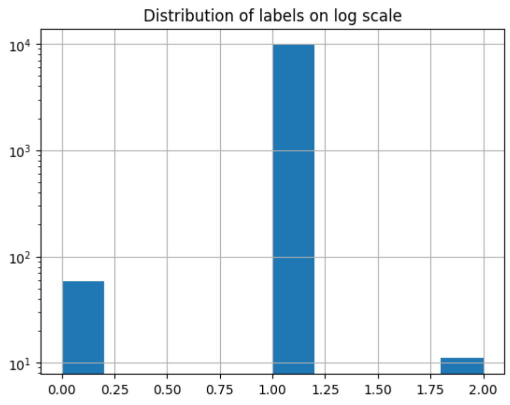


Figure 2: Balance of the classes on the original dataset. Here the histogram is on the log scale.

After that we will create three balanced classes, meaning the number of sentences in each class should be one-third of the sentences in the corpus.

In both tasks, we measured the overall accuracy and the accuracy of each class, as well as measure the sensitivity and specificity of each of the three classes.

The model for classification is given below

```
BertForClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30522, 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)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (classifier): Linear(in_features=768, out_features=3, bias=True)
)
```

Listing 2: BertForClassification model structure

5.1 Discussion

We used the cross entropy loss for the model training. For the training we used the adam optimizer and a batch size of 16. We only trained the model for 3 epochs due computational constraints.

For the unbalanced case we splitted the dataset with the same proportions of the regression model: 6000 samples for training, 2000 for validation and 2000 for testing.

For the balanced dataset we first found out the classes that has the least sample considering all the dataset. After that we choose to split the data on the following way 219 train examples, 73 validation examples, 74 test examples.

On the 3 and 2 we can see a summary of the results.

Table 2: Classification Results - Balanced dataset

Metric	Class 1	Class 2	Class 3
Total Accuracy		0.7837837837837838	
Per Class Accuracy	0.74193548	0.89473684	0.75
Sensitivity (Recall) per Class	0.74193548	0.89473684	0.75
Specificity per Class	0.82142857	0.70833333	0.8181818

Table 3: Classification Results - Unbalanced dataset

Metric	Class 1	Class 2	Class 3
Total Accuracy		0.995	
Per Class Accuracy	0.77777778	0.99648241	0.0
Sensitivity (Recall) per Class	0.77777778	0.99648241	0.0
Specificity per Class	0.46666667	0.99899244	nan

The provided results indicate the performance of a BERT-based classification model across two distinct datasets:

Balanced dataset Analysis:

Total Accuracy: Approximately 78.38%, indicating a relatively high overall correctness. Per Class Accuracy and Sensitivity (Recall): Moderate to high for all classes, with Class 2 (89.47%) performing the best. Specificity: Reasonably high across all classes, particularly for Classes 1 and 3.

Unbalanced dataset analysis:

Total Accuracy: Extremely high at 99.5%. Per Class Accuracy: Varies significantly; near-perfect for Class 2 (99.65%), moderate for Class 1 (77.78%), and zero for Class 3. Sensitivity (Recall): Mirrors accuracy trends; 0% for Class 3 is a notable concern. Specificity: High for Class 2 (99.90%), moderate for Class 1, and not available for Class 3. Overall Implications:

The balanced dataset demonstrates a balanced performance, with Class 2 as the standout. The unbalanced one shows excellent results for Class 2 but poor for Class 3, due to class imbalance.

6 Conclusions

On this article we illustrated a possible way to fine tune a BERT model for Portuguese language for 2 different tasks, one being regression and the other classification. Overall, the model showed nice performance along the tasks.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.