Natural Language Processing

Final Project Assignment

Sentiment Analysis with Deep Learning using BERT

Irimia Elena-Larisa

November 29, 2022

1 Problem statement

This project aims to apply the task of Sentiment Analysis to the Romanian language using a Deep Learning approach. More exactly, we will use a BERT model and we will fine-tune it to obtain better results.

Sentiment Analysis, also referred to as contextual mining, is an approach to *Natural Language Processing* (NLP) discipline that identifies and extracts subjective information in source material from the emotional tone behind a body of text. The story of the Sentiment Analysis is a quite recent one. This task came to the attention of the researchers in the latest times when they started to create different datasets employing film reviews, company experiences or opinions about published papers, and to develop the *Bert Base Model for Romanian*.

2 Proposed solution

The proposed solution aims to enhance the BERT model, by applying fine-tuning.

2.1 Theoretical aspects

The core of this project is the *BERT* model. It stands from *Bidirectional Encoder Representations from Transformers* and it is a recent technical innovation in applying the bidirectionality in transformers, in language modeling. It has been pre-trained on Wikipedia and BooksCorpus and it requires task-specific fine-tuning in order to be adjusted to a certain domain.

BERT uses ideas from Transformer which is an attention mechanism used for learning contextual relations between words. The transformer consists of an encoder that reads the text input and a decoder that produces a final result. From this mechanism, BERT uses only the encoder part, because its goal is to generate a language model.

The pre-training part of BERT is the key part. Two tasks are performed here. The first one is Masking and it consists of randomly choosing 15% of tokens and replacing them with the *[MASK]* token (80% of the time), a random token (10% of the time), or the same token (the token is unchanged in 10% of the time). The model then tries to predict the hidden token, based on the context. To perform this task we need to feed the model with the embeddings, let the encoder perform the computations, and add a classification layer on the top of the encoder to predict the results.

BERT takes the output of the encoder and uses that with training layers which apply an innovative training technique, *Next Sentence Prediction* (NSP). This task consists of a binary classification which told us if a pair of sentences are connected. The beginning of the sentence is marked with *[CLS]* and the text separators are marked with *[SEP]*. During the training process 50% of input are valid pairs and 50% are random pairs.

These are ways to unlock the information contained in the BERT embeddings to get the models to learn more information from the input. BERT gets the Transformer encoder to try and predict hidden or masked words. By doing this, it forces the encoder to try and learn more about the surrounding text and be better able to predict the hidden or masked word. Then, for the second training technique, it gets the encoder to predict an entire sentence given the preceding sentence.

To use BERT for Sentiment Analysis is quite easy. We only need the output vectors of each token. We use these vectors to feed the softmax layer that predicts the label.

In this project, we use a particular BERT model called *Bert Base Cased*. This model was pre-trained on Romanian language using a Masked Language Modeling (MLM) objective. This model is case sensitive and this means that it makes a difference between romanian and Romanian. We will use Bert Base Cased model because it is easier to integrate it into our algorithm.

2.2 Application

The applications of Sentiment Analysis are various. It can be used in social analysis, emotion detection or spam text classification. Our application is very simple to use because we employed a jupyter notebook in which we performed the fine-tuning process.

3 Implementation details

More implementation details can be found in the second PDF file, which is the source code of the NLP task. There we explained each step and also provided the result of every computation. Figure 1 presents the flowchart of the purposed approach and puts under the spotlight the most important steps.

The training part was performed in Google Colab using GPU and CUDA. We use PyTorch for the Sentiment Analysis task. Installing the Transformers package from Hugging Face gave us a PyTorch interface for working with BERT.

Our contribution in this project consists of the fine-tuning part of the algorithm. Also, employing a dataset containing product and movie reviews in

Romanian is another important aspect, because this dataset is not quite popular.

4 Experiments and results

For the experimental part, we tried to prepare data, to run model inference and also to visualize the obtained results. The process implies a full fine-tuning. We add some weight decay as regularization to the main weight matrices. We use a dropout layer for some regularization and a fully-connected layer for our output. In order to get the predicted probabilities from our trained model, we apply the softmax function to the outputs.

Table 1 and Table 2 present the results of applying the fine-tuning approach. As it can be seen in Figure **??**, the results of the approach tends to increase to each step. In the following, we present the source code of the proposed approach. Futhermore, the results of the sentiment analysis on some reviews can be seen in Figure 4 and Figure 5.

```
class RoSentDataset(Dataset):

  def __init__(self, reviews, targets, tokenizer, max_len):
    self.reviews = reviews
    self.targets = targets
    self.tokenizer = tokenizer
    self.max_len = max_len

  def __len__(self):
    return len(self.reviews)

  def __getitem__(self, item):
    review = str(self.reviews[item])
    target = self.targets[item]

    encoding = self.tokenizer.encode_plus(
      review,
      add_special_tokens=True,
      max_length=self.max_len,
      return_token_type_ids=False,
      pad_to_max_length=True,
      return_attention_mask=True,
      return_tensors='pt',
    )

    return {
      'review_text': review,
      'input_ids': encoding['input_ids'].flatten(),
      'attention_mask': encoding['attention_mask'].flatten()
    ,
      'targets': torch.tensor(target, dtype=torch.long)
    }
```

```
class SentimentClassifier(nn.Module):

  def __init__(self, n_classes):
    super(SentimentClassifier, self).__init__()
```

```
    self.bert = BertModel.from_pretrained(
    PRE_TRAINED_MODEL_NAME)
    self.drop = nn.Dropout(p=0.3)
    self.out = nn.Linear(self.bert.config.hidden_size,
    n_classes)

  def forward(self, input_ids, attention_mask):
    _, pooled_output = self.bert(
      input_ids=input_ids,
      attention_mask=attention_mask,
       return_dict=False
    )
    output = self.drop(pooled_output)
    return self.out(output)
```

```
EPOCHS = 5

optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=
    False)
total_steps = len(train_data_loader) * EPOCHS

scheduler = get_linear_schedule_with_warmup(
  optimizer,
  num_warmup_steps=0,
  num_training_steps=total_steps
)

loss_fn = nn.CrossEntropyLoss().to(device)
```

Epoch	Train loss	Train accuracy	Validation loss	Validation accuracy
1	0.2812	0.8906	0.2266	0.9152
2	0.1394	0.9584	0.2543	0.9297
3	0.0701	0.9825	0.3379	0.9308
4	0.0364	0.9918	0.4087	0.9297
5	0.0226	0.9951	0.4306	0.9319

Table 1: Fine-tuning approach

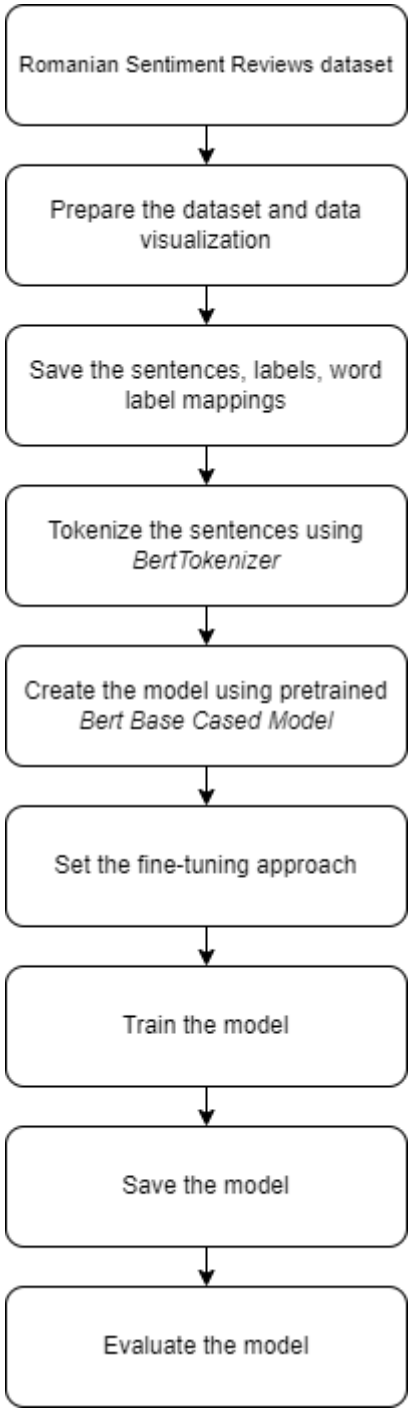

Figure 1: Flowchart of the purposed approach

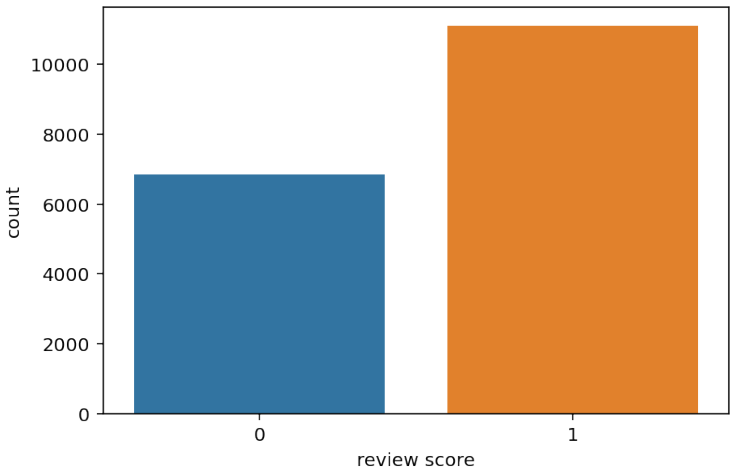

Figure 2: Fine-tuning approach - The dataset used

	Precision	Recall	F1 score
Negative	0.90	0.89	0.90
Positive	0.93	0.94	0.94

Table 2: Fine-tuning approach - Overview of the performance

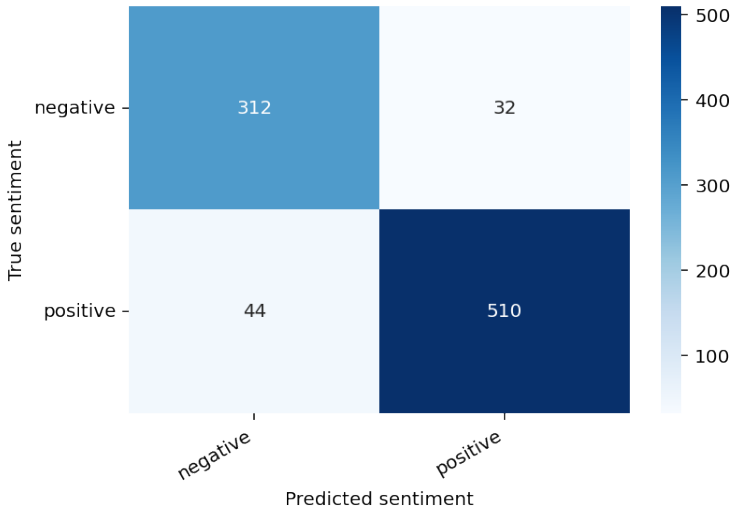

Figure 3: Fine-tuning approach - Confusion matrix

```
input_ids = encoded_review['input_ids'].to(device)
attention_mask = encoded_review['attention_mask'].to(device)

output = model(input_ids, attention_mask)
_, prediction = torch.max(output, dim=1)

print(f'Review text: {review_text}')
print(f'Sentiment  : {class_names[prediction]}')
```

```
Review text: Mi-a placut la meciul de fotbal!
Sentiment  : positive
```

Figure 4: Fine-tuning approach - Results on a positive review

```
input_ids = encoded_review['input_ids'].to(device)
attention_mask = encoded_review['attention_mask'].to(device)

output = model(input_ids, attention_mask)
_, prediction = torch.max(output, dim=1)

print(f'Review text: {review_text}')
print(f'Sentiment  : {class_names[prediction]}')
```

```
Review text: Am fost dezamagita de telefonul comandat de pe site-ul vostru!
Sentiment  : negative
```

Figure 5: Fine-tuning approach - Results on a negative review

5 Conclusion

Transformers are still a recent topic in the Natural Language Processing discipline. Furthermore, such networks have achieved more highly competitive performance than existing techniques in Sentiment Analysis task. The architectures like Transformer and BERT are paving the way for even more advanced breakthroughs to happen in the coming years. While working by myself for the project, I have learnt how to prepare data, how to run model inference and how to visualize the obtained results.

References

[1] Devlin, J., Chang, M., Lee, K., Toutanova, K., 2019. BERT: pre-training of deep bidirectional transformers for language understanding, in: Burstein, J., Doran, C., Solorio, T. (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics. pp. 4171–4186. https://doi.org/10.18653/v1/n19-1423.

[2] Dumitrescu, S.D., Avram, A., Pyysalo, S., 2020. The birth of Romanian BERT, arXiv preprint arXiv:2009.08712. https://huggingface.co/datasets/ro_sent.

[3] Transformers: State-of-the-Art Natural Language Processing, 2020. https://github.com/huggingface/transformers.