Sentiment analysis for marketing

Phase 2 submission document

Project: sentiment analysis for marketing

Content for project phase 2

Explore advanced techniques like fine-tuning pre-trained sentiment analysis models (BERT, RoBERTa) for more accurate sentiment predictions.

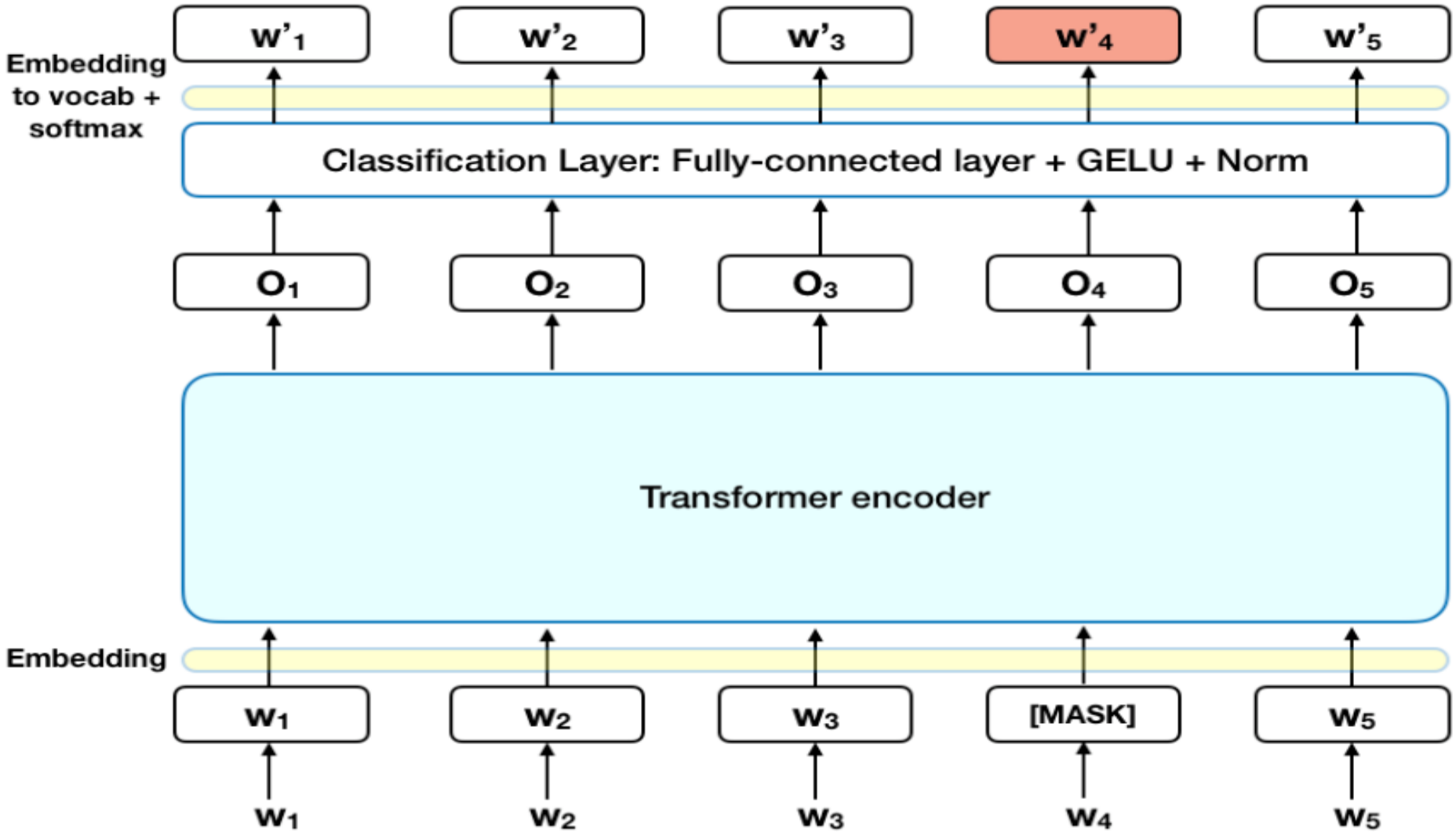
Dataset
Link: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

BERT Embeddings

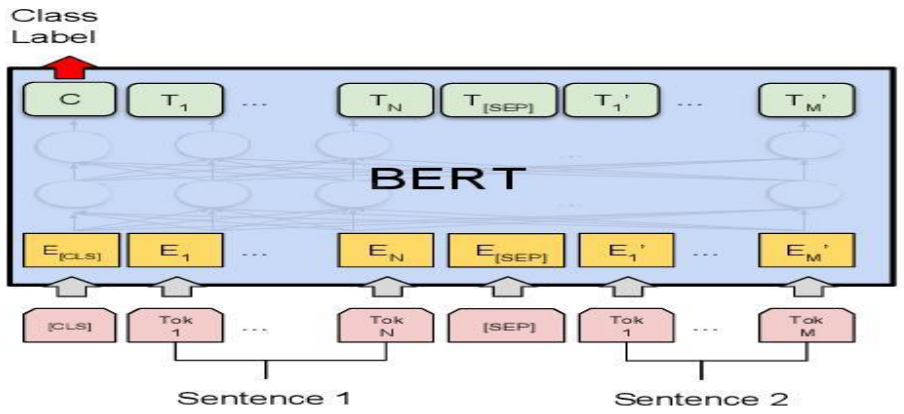
BERT is a traditional SOTA transformer architecture published by Google Research which uses bidirectional pretraining . The importance of using BERT is that it has 2 important aspects:

- Msked Language Model (MLM)
- Next Sentence Prediction(NSP)

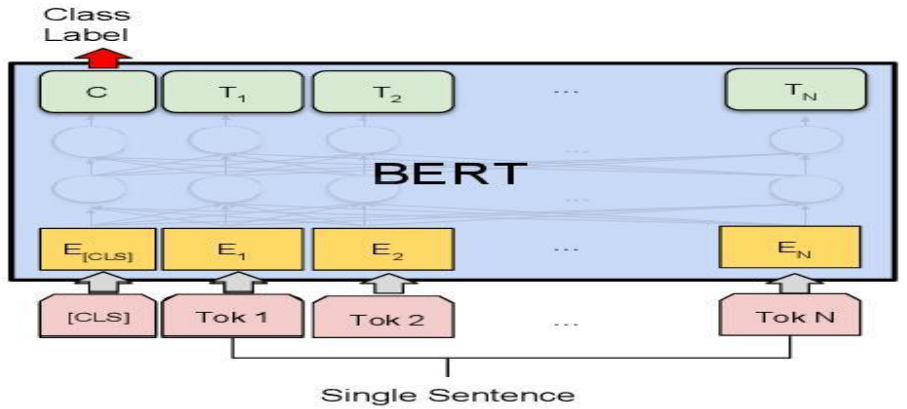
The bidirectional pre-training is essentially helpful to be used for any tasks. The Huggingface implementation is helpful for fine-tuning BERT for any language modelling task. The BERT architecture falls under an encoder-decoder(Transformer) model as follows:


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

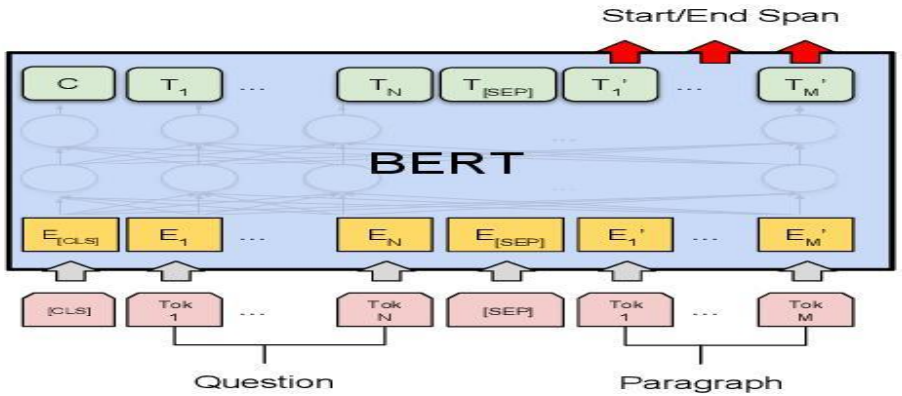
For fine-tuning and pre-training for different downstream tasks like Q/A, Classification, Language Modelling, Multiple Choice, NER etc. different layers of the BERT are used.



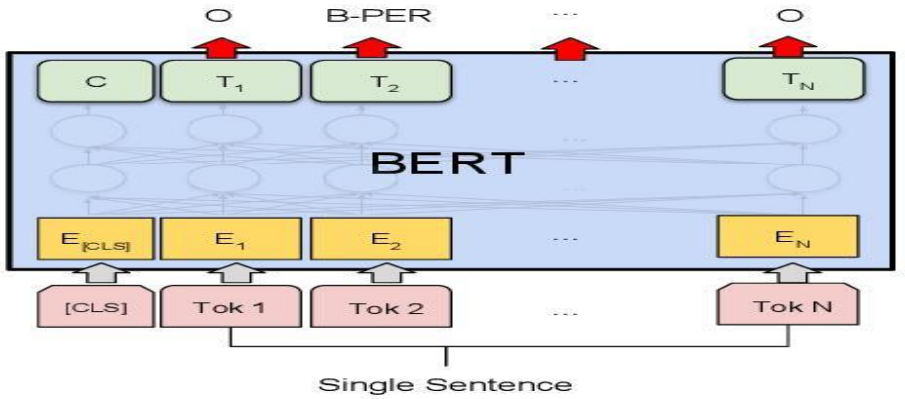
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

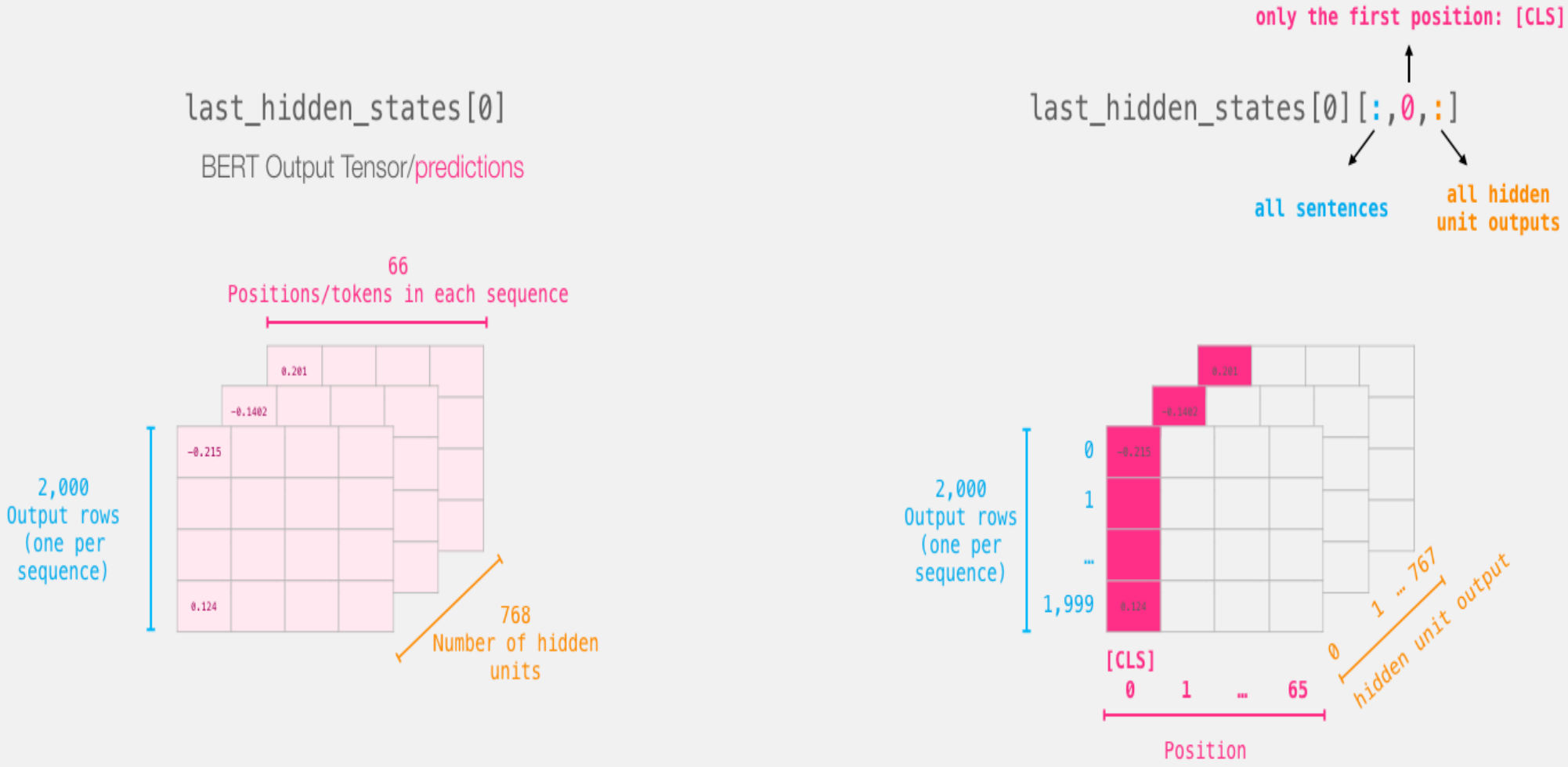
Finetuning BERT for Embeddings

For finetuning, it is to be kept in mind, there are many ways to do this. We are using BERT from Huggingfac repository while it can also be used from TF-HUB or from Google-Research repository. The reason for using HuggingFace is that the same codebase is applicable for all language models. The 3 most important input features that any language model asks for is:

- *input_ids*
- *attention_masks*
- *token_ids*

Let us first try to analyse and understand how BERT tokenizers, and model can be used in this context. The BERT documentation provides an outline of how to use BERT tokenizers and also modify it for downstream tasks.

Generally by virtue of transfer learning through weight transfer, we use pretrained BERT models from the list. This allows us to finetune it to extract only the embeddings. Since we are using Keras, we have to build up a small model containing an Input Layer and apply the tokenized(encoded) input ids, attention masks as input to the pretrained and loaded BERT model.This is very similar to creating a very own classification model for BERT using Keras/Tensorflow, but since we will be needing only the Embeddings it is safe to extract only the sentence vectors in the last layer of the model output. In most of the cases , we will see that the dimensions of the output vector is (x,768) where x depends on the number of tokenized input features. For this we extract the [CLS] tokenized feature from the ouput to just extract the sentence embeddings.


last_hidden_states[0]
BERT Output Tensor/predictions
66
Positions/tokens in each sequence
0.201
-0.1402
-0.215
0.124
2,000
Output rows
(one per
sequence)
768
Number of hidden
units
only the first position: [CLS]
last_hidden_states[0][:,0,:]
all sentences
all hidden
unit outputs
0.201
-0.1402
-0.215
0.124
0
1
…
1,999
2,000
Output rows
(one per
sequence)
[CLS]
0
1
…
65
Position
0
1 … 767
hidden unit output

Some important resources which may be helpful:

- Blog
- Extensive Nice Blog
- Good Kernel

In [1]:

```
#tokenize and encode the inputs
import transformers from transformers import
BertTokenizer,TFBertModel tokenizer =
transformers.BertTokenizer.from_pretrained('bert-large-
uncased', do_lower_case=True) bert_model =
transformers.TFBertModel.from_pretrained('bert-large-
uncased') def bert_encode(data,maximum_length):
 input_ids = []
 attention_masks = []
```

```
for i in range(len(data)):
encoded = tokenizer.encode_plus(
    data[i],
    add_special_tokens=True,
    max_length=maximum_length,
    pad_to_max_length=True,

      return_attention_mask=True,

      )

      input_ids.(encoded['input_ids’])
      attention_masks.append(encoded['attention_mask’])
        return np.array(input_ids),np.array(attention_masks)

      train_input_ids,train_attention_masks =
      bert_encode(train_df['review'][:5],1000)
```

Some layers from the model checkpoint at bert-large-uncased were not used when initializing TFBertModel: ['mlm___cls', 'nsp___cls’]

- This IS expected if you are initializing TFBertModel 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 expected if you are initializing TFBertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model). All the layers of TFBertModel were initialized from the model checkpoint at bert-large-uncased.

If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertModel for predictions without further training.

Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`.

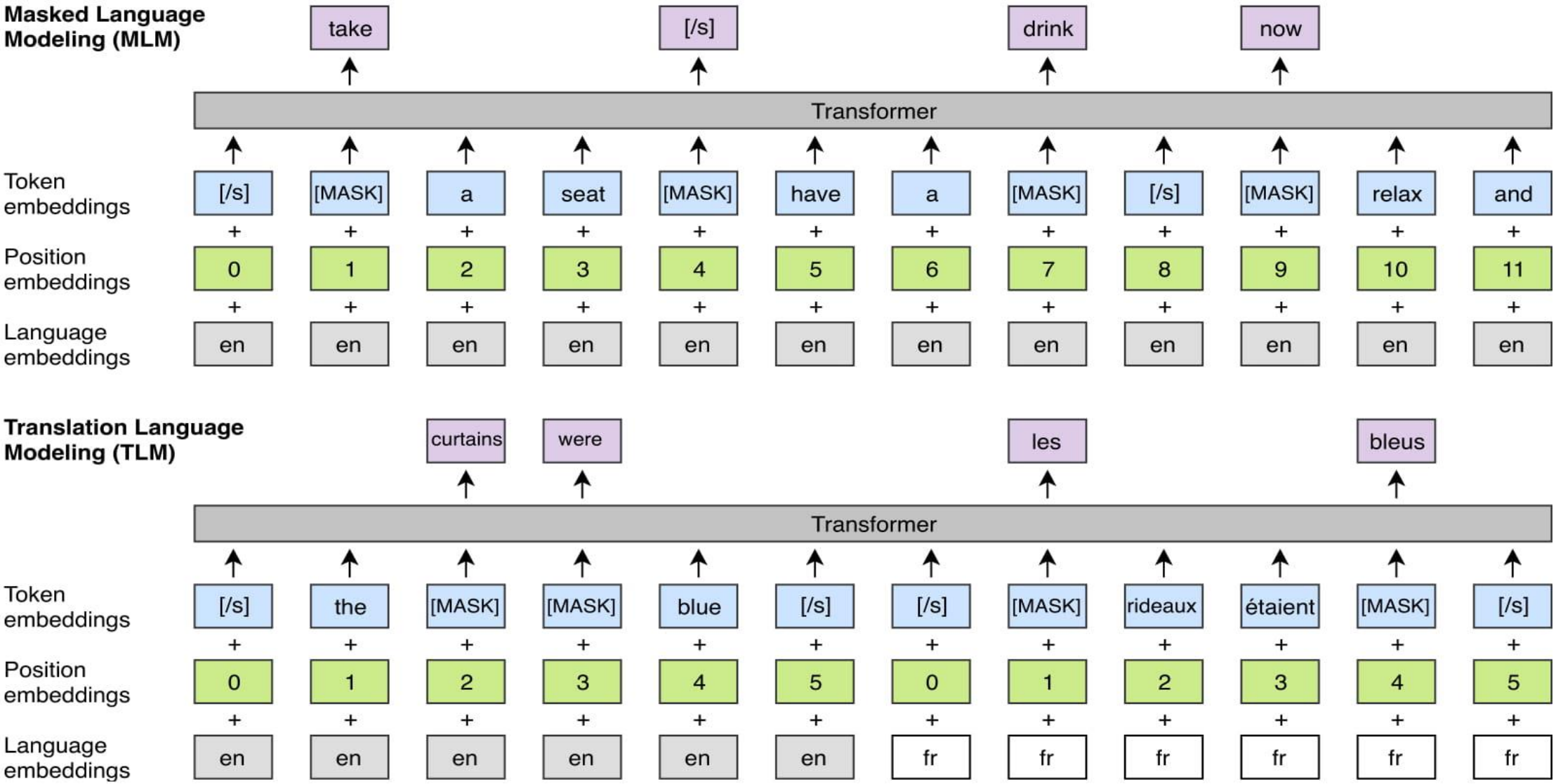
/opt/conda/lib/python3.7/site-packages/transformers/tokenization_utils_base.py:2142: FutureWarning: The `pad_to_max_length` argument is deprecated and will be removed in a future version, use `padding=True` or `padding='longest'` to pad to the longest sequence in the batch, or use `padding='max_length'` to pad to a max length. In this case, you can give a specific length with `max_length` (e.g. `max_length=45`) or leave max_length to None to pad to the maximal input size of the model (e.g. 512 for Bert). FutureWarning,

Roberta Model

- Roberta Model is a robust and large model built by Facebook Research, to alleviate undertrained nature of BERT. It trains in much larger mini-batch sizes. This provides a good model of how to train Roberta on Google cloud.The original paper can be found here, and the model architecture is provided.

Resources:

- Blog
- Blog-2


Masked Language Modeling (MLM)
take
[/s]
drink
now
Transformer
Token embeddings
[/s] [MASK] a seat [MASK] have a [MASK] [/s] [MASK] relax and
Position embeddings
0 1 2 3 4 5 6 7 8 9 10 11
Language embeddings
en en en en en en en en en en en en
Translation Language Modeling (TLM)
curtains
were
les
bleus
Transformer
Token embeddings
[/s] the [MASK] [MASK] blue [/s] [/s] [MASK] rideaux étaient [MASK] [/s]
Position embeddings
0 1 2 3 4 5 0 1 2 3 4 5
Language embeddings
en en en en en en fr fr fr fr fr fr

```
##Roberta Embeddings from transformers import AutoTokenizer, pipeline, TFRobertaModel
roberta_features1=transformer_embedding('roberta-base',z[0],TFRobertaModel)
roberta_features2=transformer_embedding('roberta-base',z[1],TFRobertaModel) distance=1-cosine(roberta_features1[0],roberta_features2[0]) print(distance)
plt.plot(roberta_features1[0]) plt.plot(roberta_features2[0]) plt.show()
```

Some layers from the model checkpoint at roberta-base were not used when initializing TFRobertaModel: ['lm_head'] - This IS expected if you are initializing TFRobertaModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).

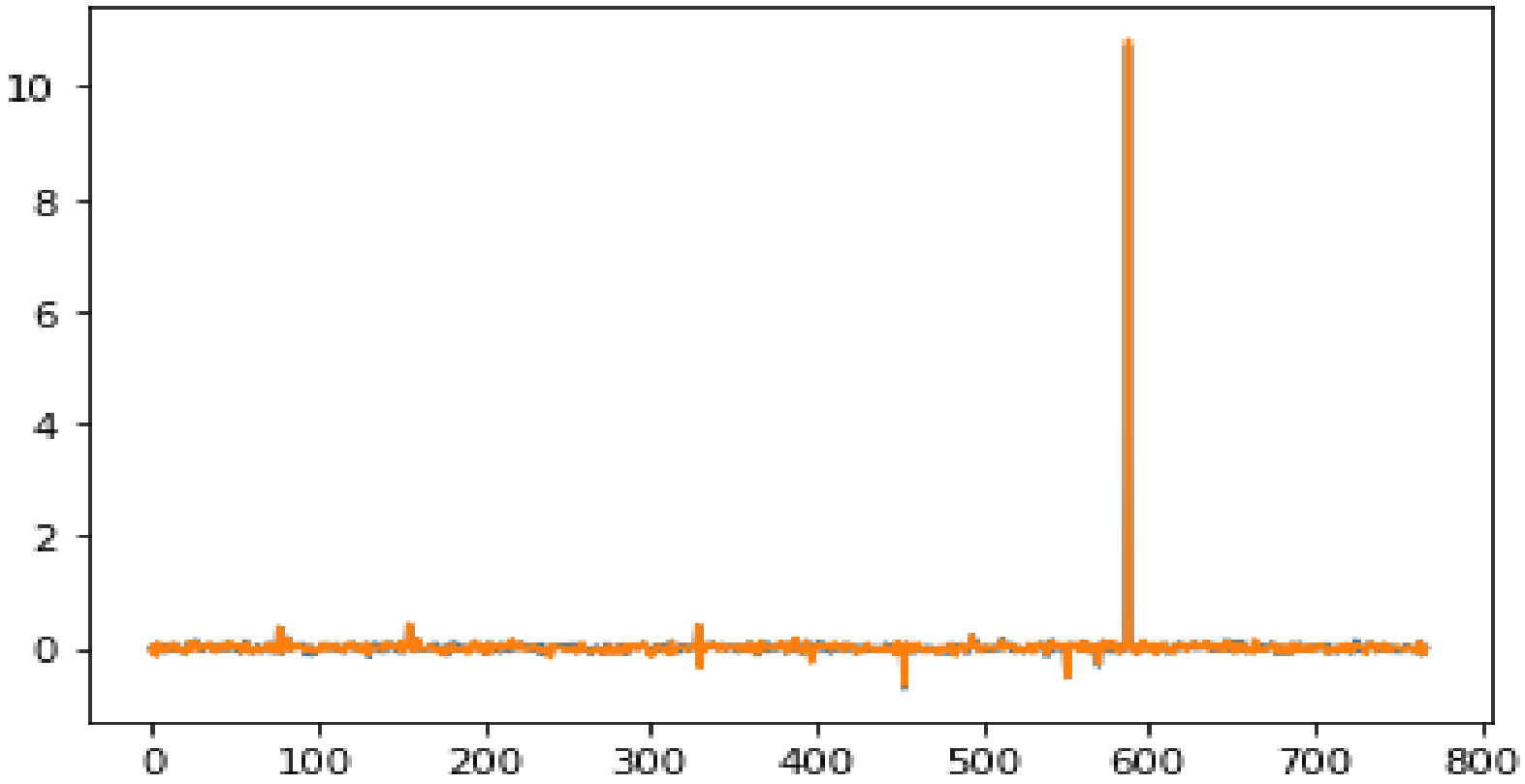
This IS expected if you are initializing TFRobertaModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).

- This IS NOT expected if you are initializing TFRobertaModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

All the layers of TFRobertaModel were initialized from the model checkpoint at roberta-base.

If your task is similar to the task the model of the checkpoint was trained on, you can already use TFRobertaModel for predictions without further training.

0.9966970127889705


10
8
6
4
2
0
0
100
200
300
400
500
600
700
800

Conclusion

- In the phase 2 conclusion, We will summarize the key findings and insights from the advanced techniques in predicting sentiment (BERT,RoBERTa).

- This Notebook has provided an idea of using a code segment for multiple Transformer based embedding models . For this it is to be kept in mind that most of these embedding models are very big architectures-Transformers - which we will be taking up in the next session on Model building.