

Sentiment Analysis with Transformer (BERT)

Computer Science Department of Shahid Beheshti University
Artificial Neural Network Course Project

Presented By : Mohammad Saeid Heidari and Shakila Jaberi

Supervisor : Dr . Katanforoush

June 2022

Outline

- Background
- Problem Definition
- Methods and Model Architecture
- Datasets
- Consider
- Computational Result
- Conclusion
- References

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Background

The **RNN** and **LSTM** neural models were designed to process language and perform tasks like classification, summarization, translation, and sentiment detection

RNN: Recurrent Neural Network

LSTM: Long Short Term Memory

In both models, layers get the next input word and have access to some previous words, allowing it to use the word's left context.

They used word embeddings where each word was encoded as a vector of 100-300 real numbers representing its meaning.

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Background

Transformers extend this to allow the network to process a word input knowing the words in both its left and right context.

This provides a more powerful context model

Transformers add additional features, like Attention, which identifies the important words in this context

And break the problem into two parts:

- An encoder (e.g., Bert)

- A decoder (e.g., GPT)

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Sentiment Analysis



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

What Is Sentiment Analysis

Sentiment analysis (or opinion mining) is a natural language processing (NLP) technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. Sentiment analysis is the process of detecting positive or negative sentiment in text. It's often used by businesses to detect sentiment in social data, gauge brand reputation, and understand customers.

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Emotion Detection Sentiment Analysis

It helps to detect and understand the emotions of the people.



Aspect based Sentiment Analysis

It is more focused on the aspects of a particular product or service.



Fine Grained Sentiment Analysis

It helps in studying the ratings and reviews given by the customers.



Intent based Sentiment Analysis

To know the intent of the customers, whether they are looking to buy the product or just browsing around, is achievable through intent analysis.



Types of Sentiment Analysis

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Sentiment analysis work-flow



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

What is BERT

BERT is an open source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context. The BERT framework was pre-trained using text from Wikipedia and can be fine-tuned with question and answer datasets. BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. (In NLP, this process is called *attention*.)

Outline

Background

Problem Definition

Model and Method

Main Idea

Datasets

Methods

Computational Result

Conclusion

What is BERT used for?

BERT is currently being used at Google to optimize the interpretation of user search queries. BERT excels at several functions that make this possible, including:

- Sequence-to-sequence based language generation tasks such as:

- Question answering
- Abstract summarization
- Sentence prediction
- Conversational response generation

- Natural language understanding tasks such as:

- Polysemy and Coreference (words that sound or look the same but have different meanings) resolution
- Word sense disambiguation
- Natural language inference
- Sentiment classification

Outline

Background

Problem Definition

Model and Method

Main Idea

Datasets

Methods

Computational Result

Conclusion

How Does BERT Work?

BERT works by three major parts :

- Architecture : Transformer
- Inputs : Text Preprocessing
- Training : Pre-training Tasks

Outline

Background

Problem Definition

Model and Method

Main Idea

Datasets

Methods

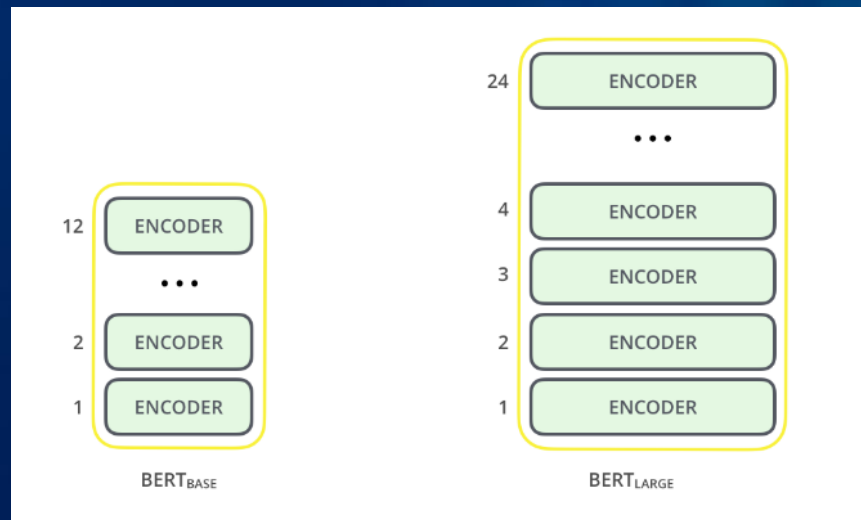
Computational Result

Conclusion

BERT Architecture

The BERT architecture builds on top of Transformer. We currently have two variants available:

- BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters
- BERT Large: 24 layers (transformer blocks), 16 attention heads and, 340 million parameters



Outline

Background

Problem Definition

Model and Method

Main Idea

Datasets

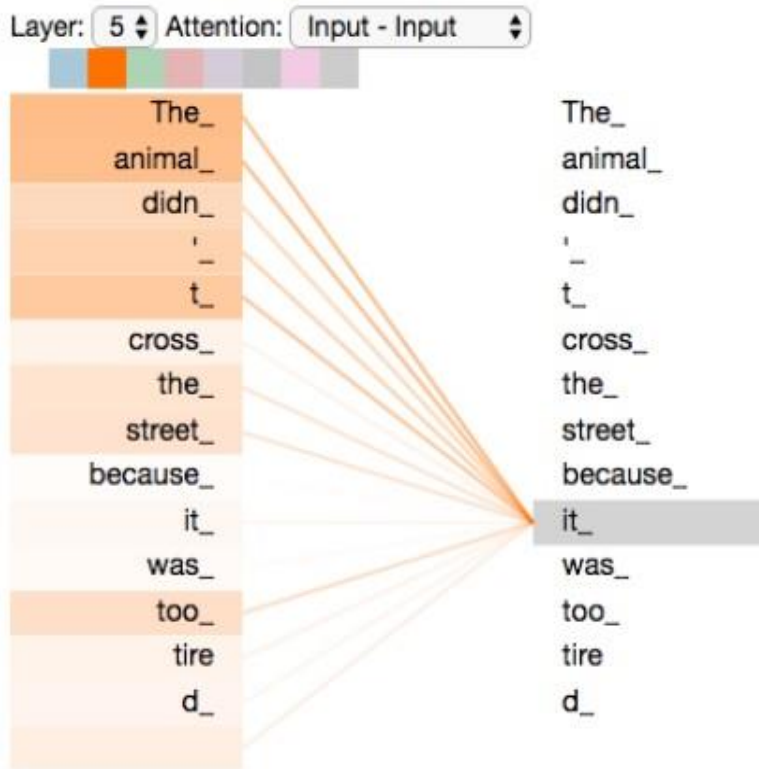
Methods

Computational Result

Conclusion

BERT Architecture

Self-attention

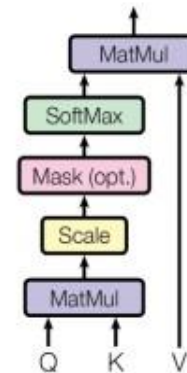


"The animal didn't cross the street because it was too tired"

"The animal didn't cross the street because it was too wide"

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Scaled Dot-Product Attention



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

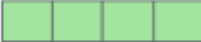
Self-attention (2)

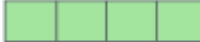
Input

Thinking


Machines

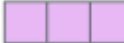
Embedding

x_1 

x_2 

Queries


q_1 

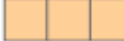
q_2 



W^Q

Keys


k_1 

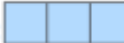
k_2 

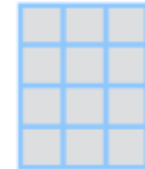


W^K

Values

v_1 

v_2 



W^V

Outline

Background

Problem Definition

Model and Method

Datasets

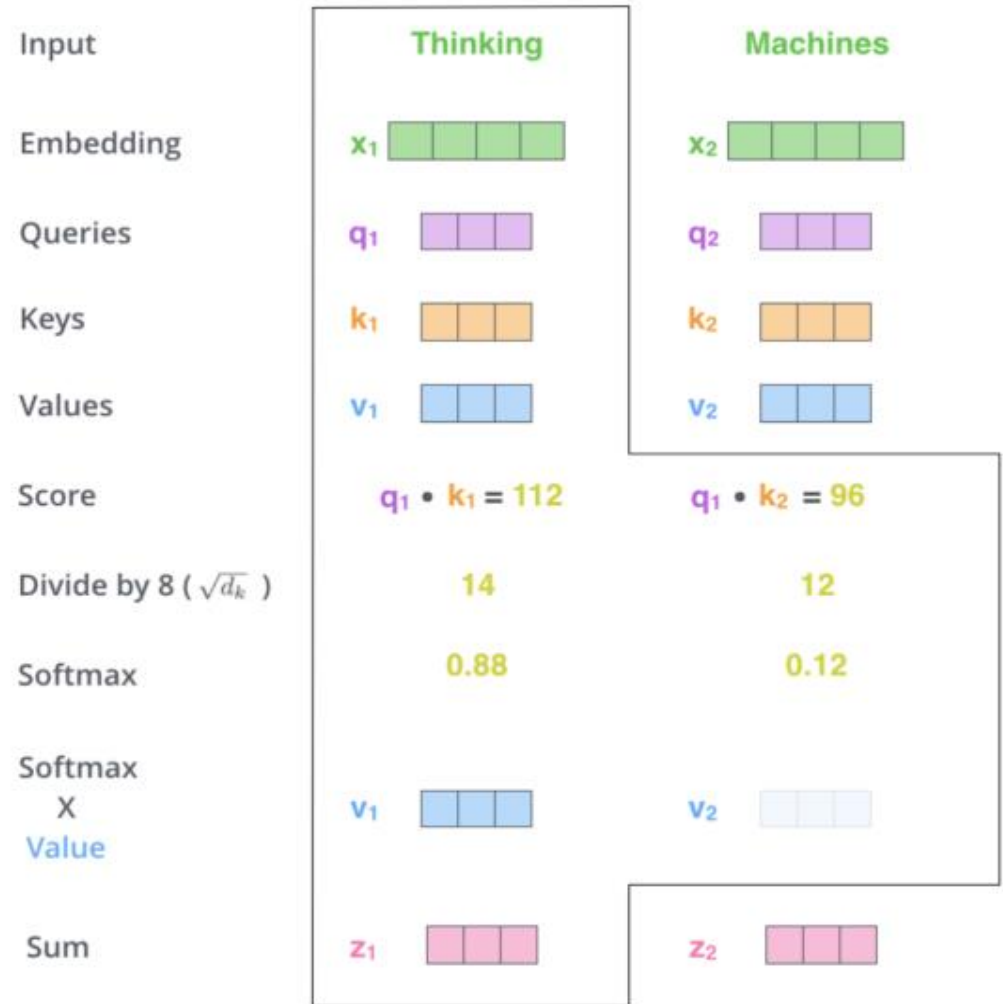
Consider

Computational Result

Conclusion

References

Self-attention (3)



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

$$X \times W^Q = Q$$

$$X \times W^K = K$$

$$X \times W^V = V$$

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Outline

Background

Problem Definition

Model and Method

Datasets

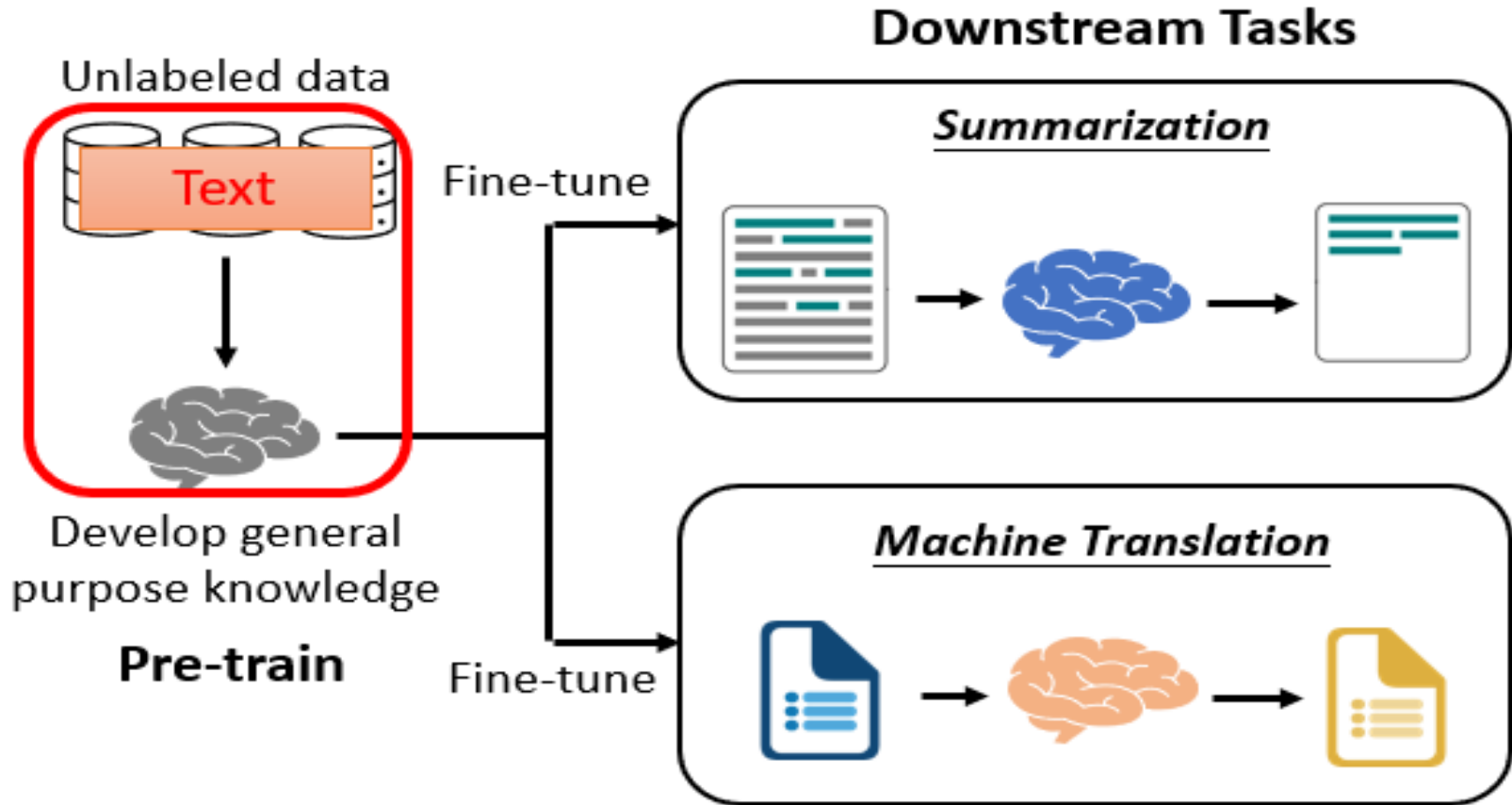
Consider

Computational Result

Conclusion

References

Bidirectional Encoder Representations from Transformers (BERT)



Outline

Background

Problem Definition

Model and Method

Datasets

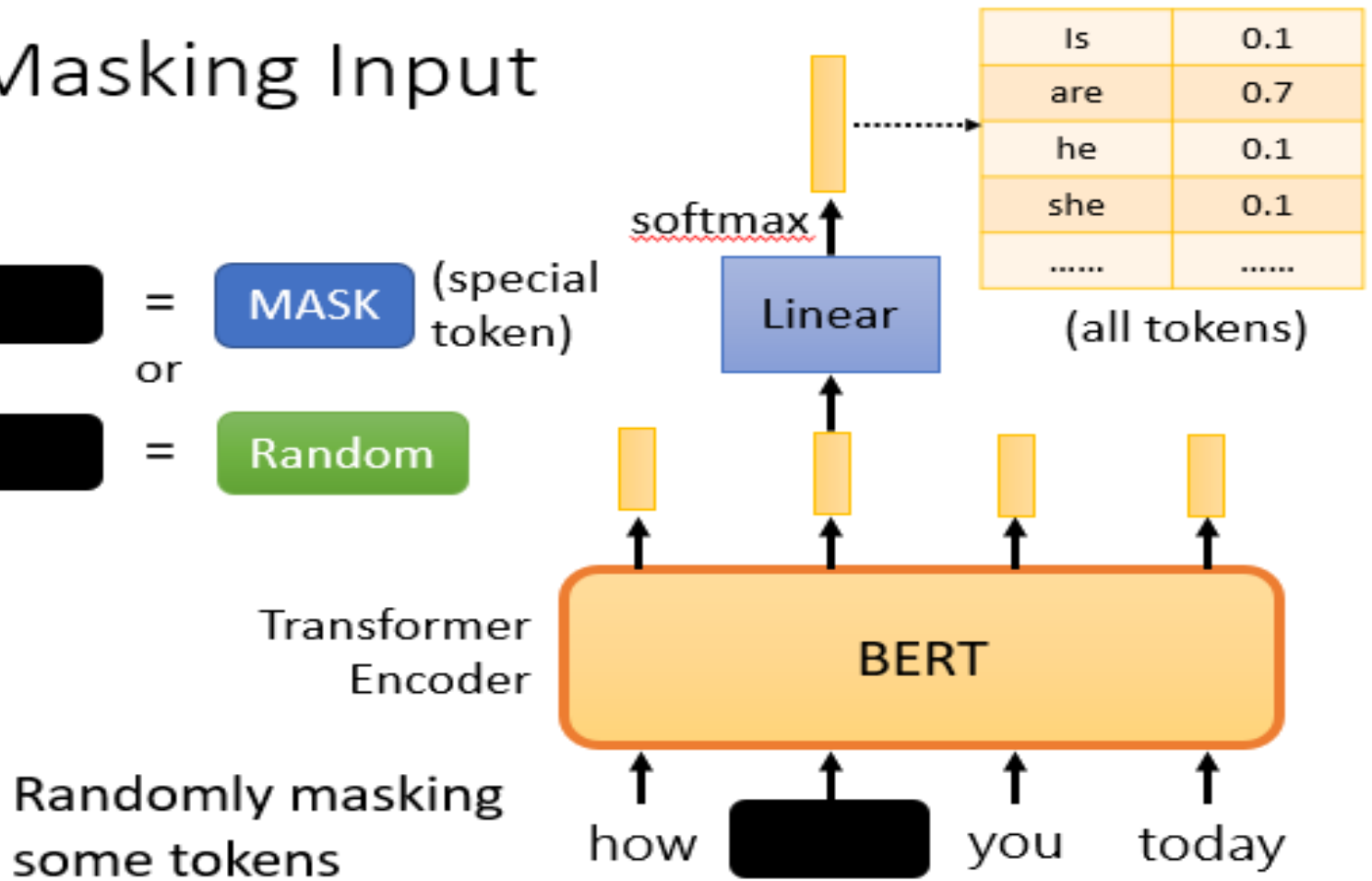
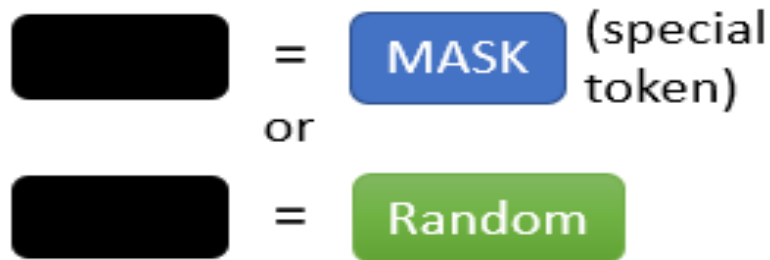
Consider

Computational Result

Conclusion

References

Masking Input



Outline

Background

Problem Definition

Model and Method

Datasets

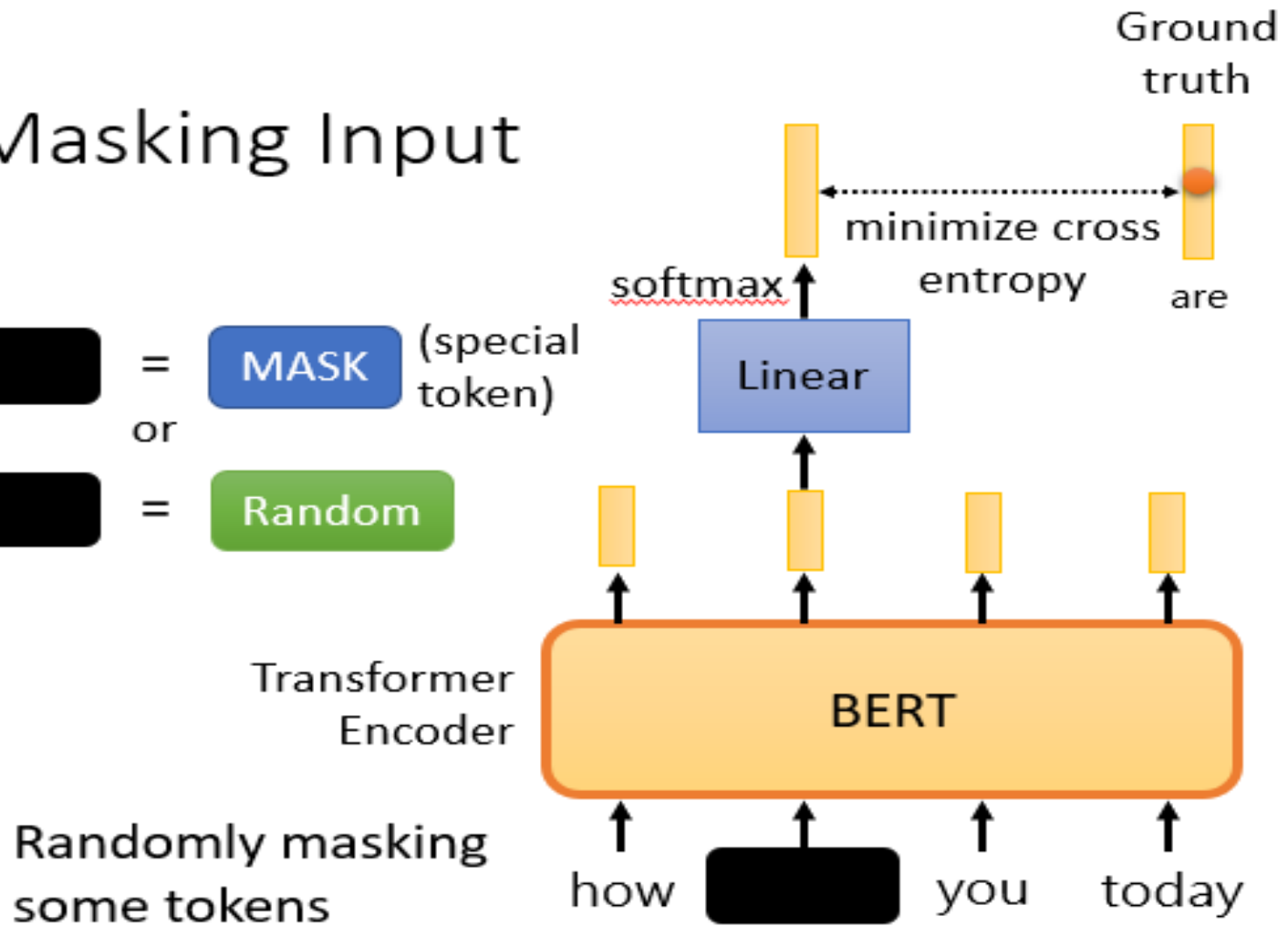
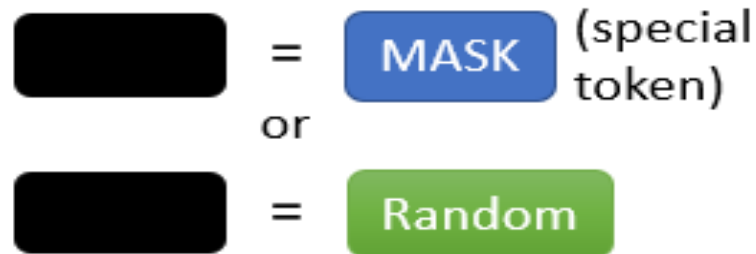
Consider

Computational Result

Conclusion

References

Masking Input



Outline

Background

Problem Definition

Model and Method

Datasets

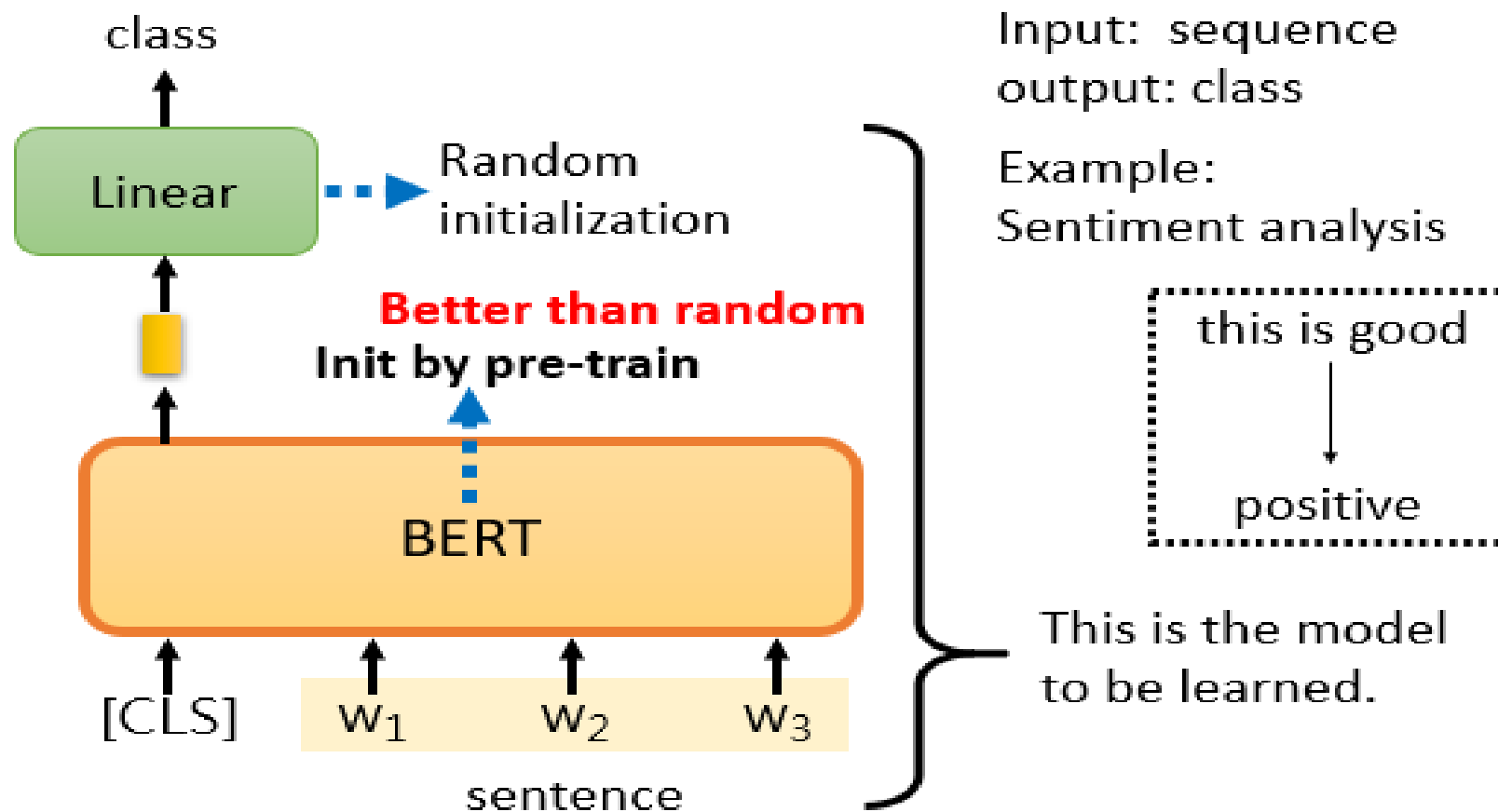
Consider

Computational Result

Conclusion

References

How to use BERT – Fintuning



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

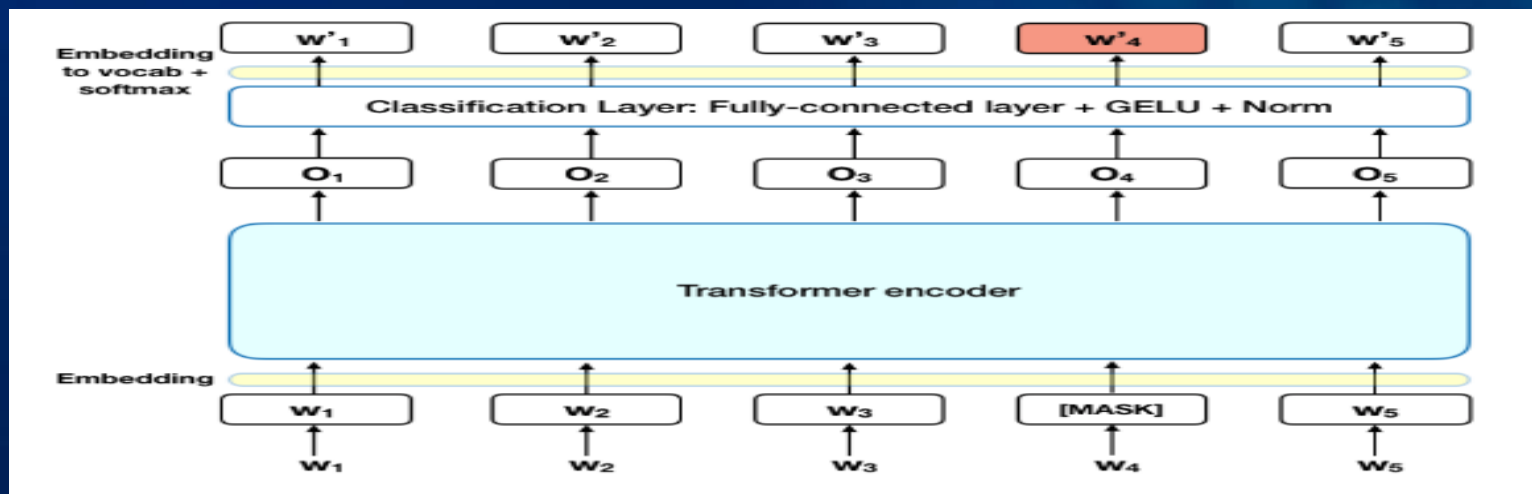
Conclusion

References

Masked LM (MLM)

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

1. Adding a classification layer on top of the encoder output.
2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
3. Calculating the probability of each word in the vocabulary with softmax.



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Next Sentence Prediction (NSP)

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.

To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
 2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2
 3. A positional embedding is added to each token to indicate its position in the sequence.
- The concept and implementation of positional embedding are presented in the Transformer paper.

Outline

Background

Problem Definition

Model and Method

Datasets

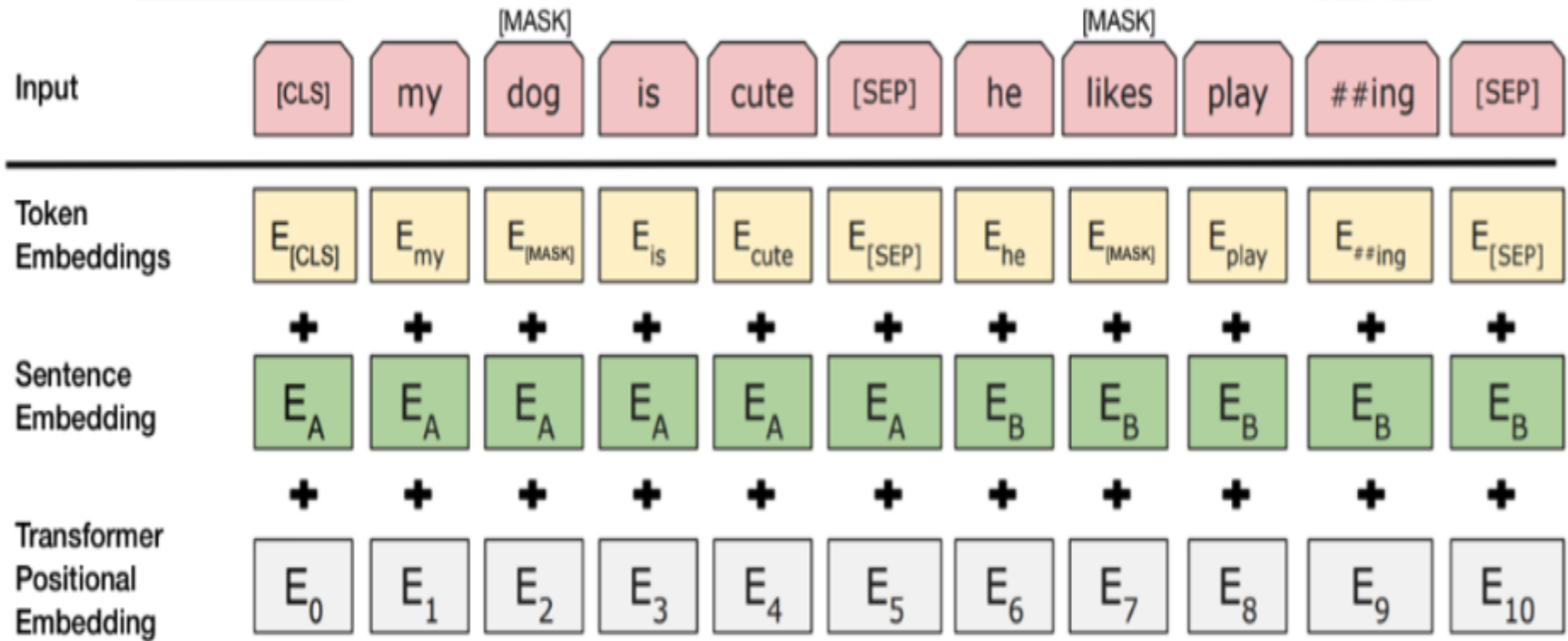
Consider

Computational Result

Conclusion

References

Next Sentence Prediction (NSP)



Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

In this project, we tested the model on two datasets

Yelp

The Yelp reviews polarity dataset is constructed by considering stars 1 and 2 negative, and 3 and 4 positive. For each polarity 280,000 training samples and 19,000 testing samples are taken randomly. In total there are 560,000 training samples and 38,000 testing samples. Negative polarity is class 1, and positive class 2. The files train.csv and test.csv contain all the training samples as comma-separated values.

Large Movie Review

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided.

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Consider Model on Large Movie Review dataset

Model compile and train

In [14]:

```
classifier_model.compile(optimizer=optimizer,  
                        loss=loss,  
                        metrics=metrics)  
  
print(f'Training model with {tfhub_handle_encoder}')  
history = classifier_model.fit(x=train_ds,  
                              validation_data=val_ds,  
                              epochs=epochs)
```

Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1

Epoch 1/5

313/313 [=====] - 181s 557ms/step - loss: 0.5252 - binary_accuracy: 0.7247 - val_loss: 0.3795 - val_binary_accuracy: 0.8270

Epoch 2/5

313/313 [=====] - 173s 554ms/step - loss: 0.3668 - binary_accuracy: 0.8378 - val_loss: 0.3755 - val_binary_accuracy: 0.8386

Epoch 3/5

313/313 [=====] - 173s 553ms/step - loss: 0.2991 - binary_accuracy: 0.8725 - val_loss: 0.3678 - val_binary_accuracy: 0.8470

Epoch 4/5

313/313 [=====] - 173s 553ms/step - loss: 0.2494 - binary_accuracy: 0.8985 - val_loss: 0.3920 - val_binary_accuracy: 0.8520

Epoch 5/5

313/313 [=====] - 174s 557ms/step - loss: 0.2145 - binary_accuracy: 0.9148 - val_loss: 0.4069 - val_binary_accuracy: 0.8510

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Consider Model on Yelp dataset

Model compile and train

```
classifier_model.compile(optimizer=optimizer,  
                        loss=loss,  
                        metrics=metrics)  
  
print(f'Training model with {tfhub_handle_encoder}')  
history = classifier_model.fit(x=train_data,  
                              validation_data=val_data,  
                              epochs=epochs)
```

Training model with https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A-8/1

Epoch 1/2

625/625 [=====] - 341s 534ms/step - loss: 0.3440 - binary_accuracy: 0.8449 - val_loss: 0.2407 - val_binary_accuracy: 0.9019

Epoch 2/2

625/625 [=====] - 328s 525ms/step - loss: 0.2226 - binary_accuracy: 0.9108 - val_loss: 0.2289 - val_binary_accuracy: 0.9081

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Result BERT on Yelp Dataset **Without** Changing Loss Function and Optimizer

Model evaluation

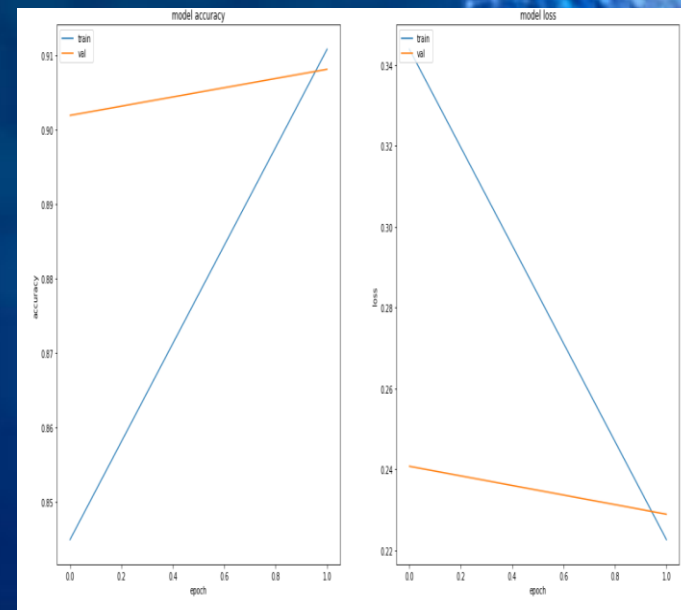
```
def round_func(n):  
    if n < 0.5:  
        return 0  
    else:  
        return 1
```

```
preds = classifier_model.predict(test_data)  
preds = preds.reshape(1, -1)[0]  
preds = np.array(list(map(lambda a:round_func(a), preds)))
```

157/157 [=====] - 49s 315ms/step

```
from sklearn.metrics import classification_report  
print(classification_report(y_test, preds, target_names=['negative', 'positive']))
```

	precision	recall	f1-score	support
negative	0.91	0.92	0.91	5367
positive	0.90	0.89	0.90	4633
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000



Outline

Background

Problem Definition

Model and Method

Datasets

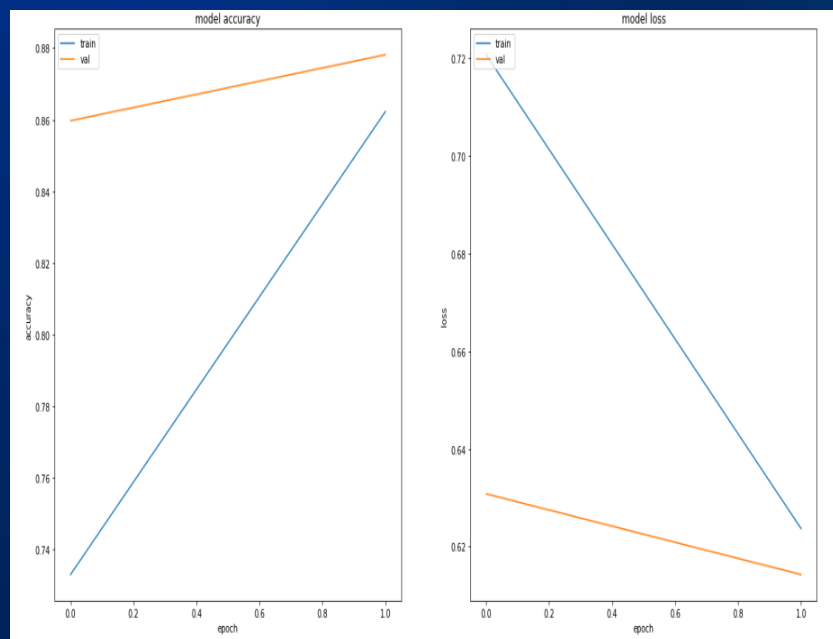
Consider

Computational Result

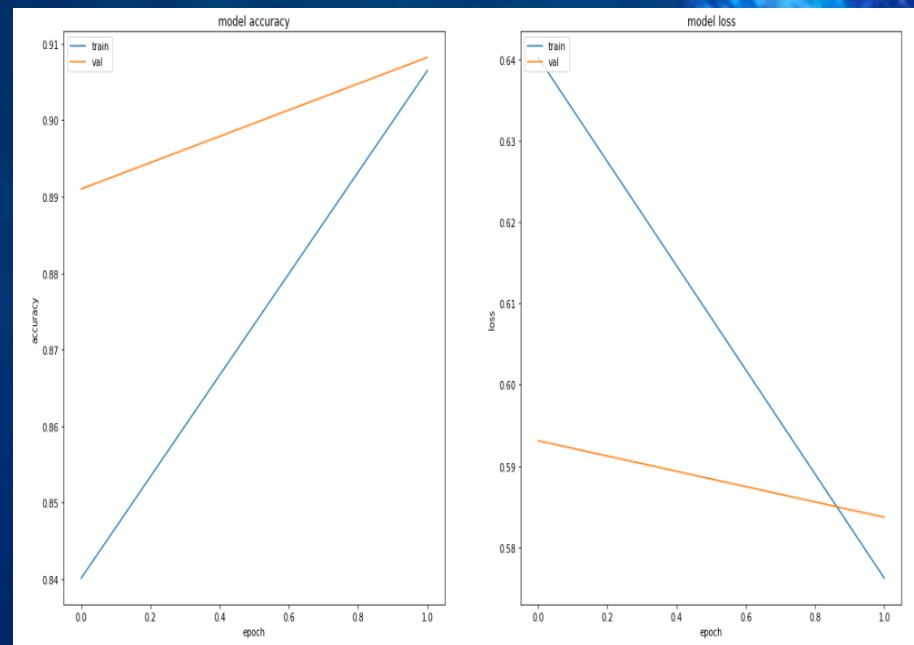
Conclusion

References

Result BERT on Yelp Dataset With Changing Loss Function and Optimizer



poisson_lamb



poisson_adam

Outline

Background

Problem Definition

Model and Method

Datasets

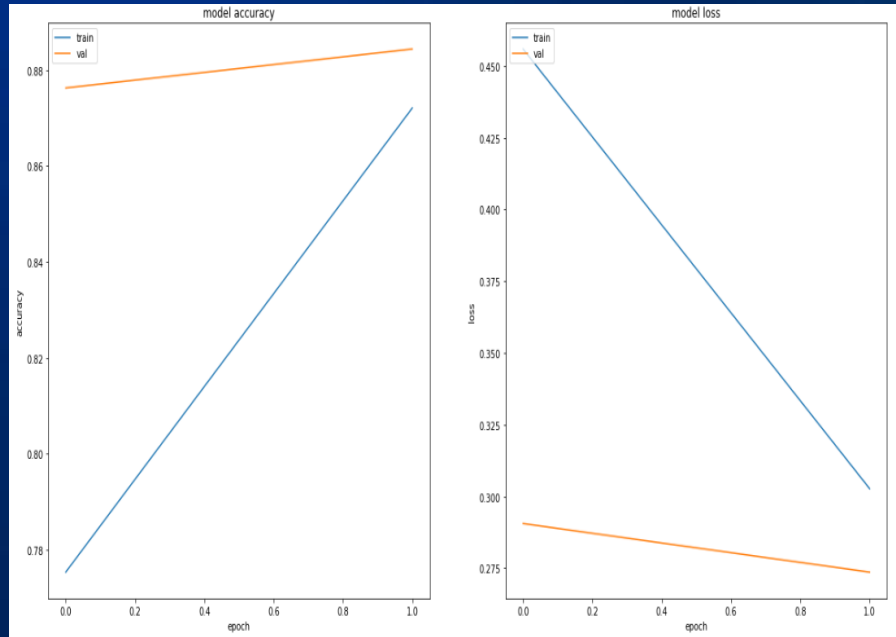
Consider

Computational Result

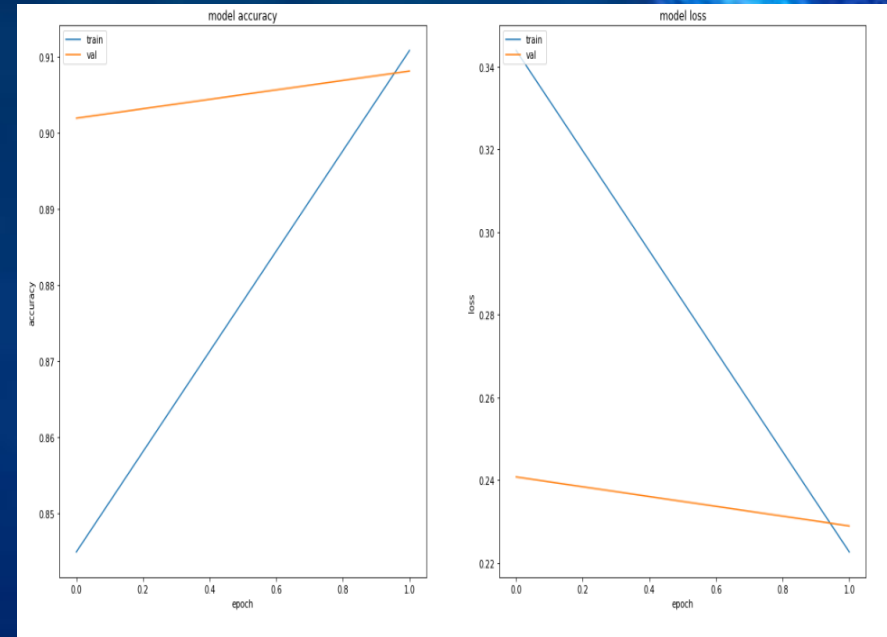
Conclusion

References

Result BERT on Yelp Dataset With Changing Loss Function and Optimizer



BinaryCrossEntropy_lamb



BinaryCrossEntropy_adam

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Conclusion

BERT is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. The fact that it's approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future.

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

References

<https://arxiv.org/abs/1810.04805>

<http://jalammar.github.io/illustrated-bert/>

<https://medium.com/mlearning-ai/twitter-sentiment-analysis-with-deep-learning-using-bert-and-hugging-face-830005bcdbbf>

<https://www.blog.google/products/search/search-language-understanding-bert/>

Outline

Background

Problem Definition

Model and Method

Datasets

Consider

Computational Result

Conclusion

References

Thank You for Attention

