# BERT Sentiment Analysis to Detect Twitter Sarcasm

# (Naive Approach)

*At the final stage of your project, you need to deliver the following:*

* *Your documented source code and test set predictions.*
* *Explain your model, and how you perform the training. Describe your experiments with other methods that you may have tried and any hyperparameter tuning.*
* *A demo that shows your code can actually run on the test set and generate your submitted predictions. You don’t need to run the training process during the demo. If your code takes too long to run, try to optimize it, or write some intermediate results (e.g. inverted index, trained model parameters, etc.) to disk beforehand.*

## Introduction

*Sarcasm is a form of figurative language that implies a negative sentiment while displaying a positive sentiment on the surface*

*(Joshi et al., 2017).*

I present a ***Naive*** approach to detect *Twitter* tweet sarcasm using a transformers-based pre-trained model that consider only the ***response***. This approach completely ignores the ***context*** of the response to train the model.

The model uses a transformer encoder to generate the embedding representation for the response. The model is trained and evaluated on the given training and testing datasets. My best performance model gives ***F1***-score of ***75.79%****, beating the baseline score after 4 epoch iterations (epoch # 4).*

***C. Dataset Description*** section of this document explains further the ***response*** and ***context*** relationship.

Important files in the project:

1. Documented software code

* Training and evaluation [(link)](https://github.com/zen030/CourseProject/blob/main/NAIVE_BERT_sentiment_analysis.ipynb)
* Evaluation of a trained model for demo purpose [(link)](https://github.com/zen030/CourseProject/blob/main/DEMO_Model_Evaluation.ipynb)

1. Best performance testing set predictions (answer.txt) [(link)](https://github.com/zen030/CourseProject/blob/main/answer.txt)
2. Best performance trained model [(link)](https://drive.google.com/file/d/1EMcBXsFPqOVg4w_-Nob4ebWA0qTr9SLQ/view?usp=sharing)
3. Training dataset [(link)](https://github.com/zen030/CourseProject/blob/main/train.jsonl)
4. Testing dataset [(link)](https://github.com/zen030/CourseProject/blob/main/test.jsonl)

## Bidirectional Encoder Representations from Transformers (BERT)

This project uses BERT; a transformer-based technique for Natural Language Processing pre-training developed by a team in *Google*.

The original English language BERT model comes with 2 pre-trained model types:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model Type | Layer | Hidden | Head | Parameter | Corpus Words |
| **Base** | 12 | 768 | 12 | 110 M | 800 M |
| **Large** | 24 | 1024 | 16 | 340 M | 2.500 M |

*Table 1: BERT original model types*

BERT Large model essentially has a better computing leverage than the base model. Google team trained the large model using larger corpus word size than the base model. The large model is expected to perform better than the base model in most of the NLP tasks such as sentiment analysis.

Original BERT paper is available here [(link)](https://arxiv.org/pdf/1810.04805.pdf).

## Dataset Description

There are two *Twitter* tweet datasets available for this project:

1. Training dataset: a labelled dataset to train the model
2. Testing dataset: tweet with ID to evaluate the trained model

For training dataset, each line contains a JSON object with the following columns:

* ***label***: **SARCASM** or **NOT\_SARCASM**
* ***response***: the classified tweet
* ***context***: the conversation context of the response

For training dataset, each line contains a JSON object with the following columns:

* ***id***: identifier for sample
* ***response***: the tweet to be classified
* ***context***: the conversation context of the response

|  |  |
| --- | --- |
| Training Dataset | Testing Dataset |
| 5000 lines | 1800 lines |

*Table 2: Dataset size statistics*

A more detail dataset description is available in the project competition Github repository [(link)](https://github.com/CS410Fall2020/ClassificationCompetition/blob/main/README.md).

## The Naive Approach

I hypothesize the ***context*** does not ***always*** support the sentiment of a ***response***. Context can have opposing affect to the response’s sentiment.

I hypothesize there are 2 types of contexts:

1. ***Positive context*** is the context that support the sentiment of a response.
2. ***Negative context*** is the context that does not support the sentiment of a response.

Sentiment

Negative

Context

*Fig.1: Illustration of context reduce sentiment quality*

Positive

Context

S e n t i m e n t

*Fig.2: Illustration of context increase sentiment quality*

To train the model it is important to select the context that support the response’s sentiment. In this project I use only the sentiment-labelled response to train the model, and I completely ignore the context. I call this a ***Naive*** approach.

*In the future project*, I can use advance machine learning techniques to utilize both response and context to train the model by ***selectively*** reconstruct the context to support the sentiment of the response.

## The model and training

BERT Model used for training:

* BERT Large uncased
* BERT Base uncased

Training parameters:

* Learning rate: 2e-5
* Epsilon: 1e-8
* Random seed value: 17
* Epochs: 4 iterations
* Environment: Google Colab PRO

|  |  |
| --- | --- |
|  | Response Max. Chars |
| Training Dataset | 315 |
| Testing Dataset | 310 |

As of today, *Twitter* tweet maximum length is ***280*** characters or ***Unicode glyphs***. Some glyphs will count as more than one character [(reference link)](https://developer.twitter.com/en/docs/counting-characters).

BERT model can handle a text with maximum 512 characters. If the input text is more that 512 characters, the model can truncate the text to 512 characters. That means we might miss important word that indicate the sentiment. Luckily response text in both training and testing datasets is less than 512 characters, we can be sure to train the model by considering all words.

Graphical user interface, text, application

Description automatically generated

*Fig.3: Source code to check maximum training and evaluation response characters length*

## Software Code

I implemented software code into 2 Google Colab Notebooks:

1. Training and evaluation notebook: *NAIVE\_BERT\_sentiment\_analysis.ipynb* [(link)](https://github.com/zen030/CourseProject/blob/main/NAIVE_BERT_sentiment_analysis.ipynb)
2. Evaluate selected trained-model notebook (for DEMO purpose): *DEMO\_Model\_Evaluation.ipynb* [(link)](https://github.com/zen030/CourseProject/blob/main/DEMO_Model_Evaluation.ipynb)

Environment Google Colab PRO.

1. *NAIVE\_BERT\_sentiment\_analysis.ipynb*
2. Colab Configuration
3. Python module installation

Text

Description automatically generated

1. Copy train.jsonl and test.jsonl files from Google Drive to Colab session

I have already copied train.json and test.jsonl files to my Google Drive account created for this project. I have shared the files to public. The following code will copy the files to the current running Colab session:

Graphical user interface, text, email

Description automatically generated

This source code will prompt you to a URL. Login to your Google account, click the link to get a code and paste the code to the “Enter verification code:” text box, then press “Enter” key.

Graphical user interface, text, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

At this point have our training and testing datasets in the current running Colab session!

1. Mounting Google Drive to Colab session (To save result files)

Graphical user interface, text, application

Description automatically generated

Follow the authentication process as described in the step above, when prompted, to mount Google Drive to the current Colab session. This allow us to have access to Google Drive directory where we will store our result files. In this project, I mounted my Google Drive directory to ‘***./content/uiuc***’ folder.

A picture containing text

Description automatically generated

*Mounted Google Drive in Colab Session where to keep result files permanently*

1. The main Python Class

In this project I implemented a Python class that handles the following tasks:

* + Read the dataset from jsonl files into a list of json
  + Convert list of json to Pandas DataFrame
  + Create the BERT Model
  + Run the training and save the model for each epoch
  + Evaluate the model and store the result into file

Below is the class signatures (for details source code please check it here [link](https://github.com/zen030/CourseProject/blob/main/NAIVE_BERT_sentiment_analysis.ipynb))

Text, letter

Description automatically generated

## Training and Evaluation experiments

To test my hypothesis, I run only 2 experiments with the same hyperparameters. I used 2 BERT models:

* + BERT base uncased
  + BERT LARGE uncased

Hyperparameters:

* + Learning rate: 2e-5
  + Epsilon: 1e-8
  + Seed value for random: 17
  + EPOCH: 4 iterations

1. Experiment-1: BERT base uncased

Text

Description automatically generated

1. Experiment-2: BERT LARGE uncased

Text

Description automatically generated

This will generates the files in the current Colab session folder as shown below:

Graphical user interface, text, application

Description automatically generated

1. Save the result files to Google Drive

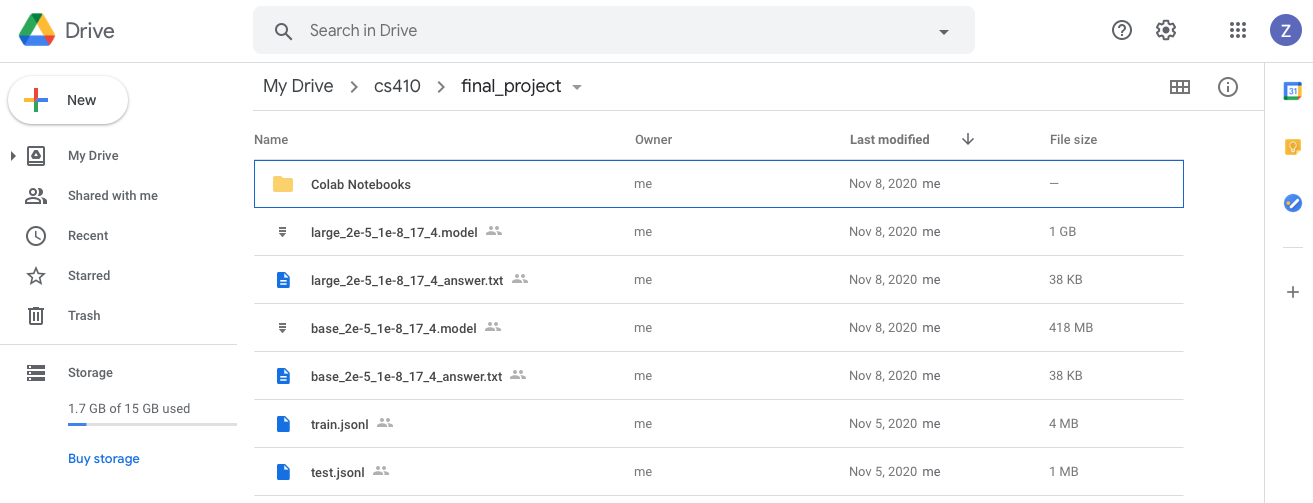
The files in Colab session will be deleted when Colab session is ended. We need to store the files permanently in other location, in my case, in the project I use Google Drive.

The code below will store the result files to my Google Drive folder using the folder mounted in the earlier step. In this project, I only store EPOCH # 4 model and evaluation result.





The files are copied to the Google Drive folder:



We will use the files from Google Drive to run a DEMO which I will illustrate in the following section.

1. *DEMO\_Model\_Evaluation.ipynb*

In the demo, I will demonstrate how to generate ‘answer.txt’ from the BERT LARGE uncased trained model. The trained model is available to the public here [link](https://drive.google.com/file/d/1EMcBXsFPqOVg4w_-Nob4ebWA0qTr9SLQ/view?usp=sharing). With this model, we will create the reproduce evaluation result which is available to the public here [link](https://drive.google.com/file/d/1b2vKQDbo-BbLwWok9qDeZzkAxpQYfaSp/view?usp=sharing).

The first 3-steps are already explained in the previous section:

1. Colab configuration
2. Copy result files from Google Drive to Colab session, files copied are:
   1. Large\_2e-5\_1e-8\_17\_4.model
   2. test.jsonl
3. Prepare Panda DataFrame for the testing dataset
4. Evaluate the model using testing dataset
   1. Load the model from the trained-model file
   2. Create the tokenizer to encode testing dataset
   3. Create the data loader to run the evaluation batch iterations

Graphical user interface, text, application, email

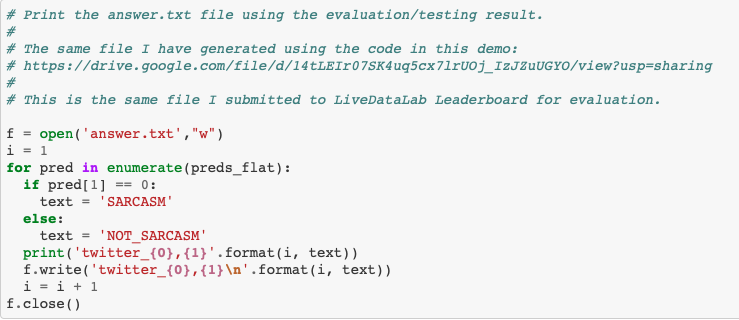
Description automatically generated

1. Run the evaluation batch iteration

Text

Description automatically generated

1. Generate the ‘anwer.txt’ file



1. Post ‘answer.txt’ to LiveDataLab for scoring

Table

Description automatically generated

*Leaderboard snapshot on 03-Nov-2020*

***Result and Conclusion***

|  |  |  |  |
| --- | --- | --- | --- |
| Model | F1-Score | Recall | Precision |
| BERT Large uncased | **0.757905138339921** | 0.8522222222222222 | 0.6823843416370107 |
| BERT Base uncased | 0.7458777885548012 | 0.8544444444444445 | 0.6617900172117039 |

Surprisingly, Base model performs almost as good as Large model.

In this project, I did try to use different trained model such as RoBERTa and XLNet (and different hyperparameters), but for a reason I could not produce result higher than BERT Large uncased score. In this project as proposed in the project proposal, I am reporting the result for BERT model only.

In the future, I would like to explore more on the following topics:

* Using advance machine learning technique to evaluate hyperparameters. I use the original BERT paper [(link)](https://arxiv.org/pdf/1810.04805.pdf) to choose hyperparameters for my experiments.
* Utilizing the context instead of using response only. I hypothesize that context if handles properly can be used as an additional dataset to train the model.
* Explore another model such as RoBERTa and XLNet.

***Reference***

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Car- man. 2017. Automatic Sarcasm Detection: A Survey. *ACM Computing Surveys*, 50(5):1–22.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.