## CS 410: Text Information Systems Final Report

**Team Information:** NLP AI Innovators

1. Elias Amuneke

# Project Overview:

# The objective of this project was to implement a sentiment analysis competition using the Twitter Tweets Sentiment classification dataset available on Kaggle. The dataset comprises tweets labeled with negative, positive, or neutral sentiments. The goal was to train a high-performance model for sentiment classification, focusing on the context within the Twitter text. The project utilized the BERT (Bidirectional Encoder Representations from Transformers) model, a state-of-the-art natural language processing (NLP) model, for the sentiment analysis task. The BERT model was loaded using the transformers library, with the 'bert-base-uncased' configuration. The sentiment labels were categorized into three classes: negative, positive, and neutral. The dataset was preprocessed by splitting it into training and testing sets using the train\_test\_split function from scikit-learn. The BERT tokenizer was employed to convert the raw text into tokenized sequences suitable for model input. The data was organized into a custom dataset class, TweetDataset, and DataLoader instances were created for both the training and testing sets. The model was fine-tuned using the AdamW optimizer with a learning rate of 5e-5 over a specified number of epochs. The training process was logged for time tracking, and the model's performance was evaluated on the testing set. Classification metrics such as accuracy and a detailed classification report were calculated to assess the model's effectiveness in predicting sentiment.

The competition framework encourages participants to develop and fine-tune models to achieve the highest accuracy on sentiment classification. The project leverages BERT's powerful contextual understanding of language to capture nuanced sentiment in the context of Twitter text. The implementation demonstrates a systematic approach to handling NLP tasks with pre-trained models, providing a foundation for participants to explore advanced techniques for improving sentiment analysis on social media data. Overall, this project fosters a competitive and collaborative environment, driving innovation in sentiment analysis within the context of Twitter

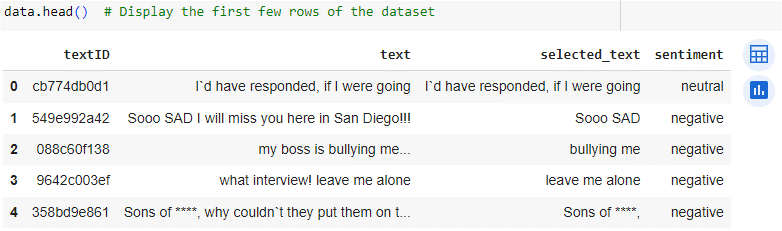
<https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset>. We competed to achieve the highest model performance.

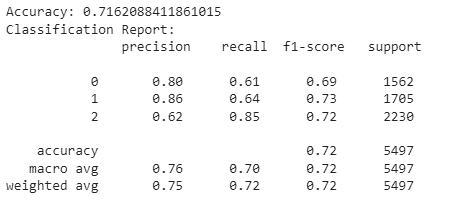
# Implementation Documentation:

The primary codebase for this project resides in the tweet\_classifier\_model.py file, serving as a comprehensive script for training and evaluating a sentiment analysis model on the Twitter Tweets Sentiment classification dataset.

1. **Data Loading and Pre-processing:** The initial steps involve importing essential libraries such as transformers, torch, and pandas. The dataset is loaded from a CSV file using pandas, specifically the 'archive/Tweets.csv' file. The dataset is then split into training and testing sets using the train\_test\_split function from scikit-learn. The selected columns, 'selected\_text' for training and 'text' for testing, are chosen for sentiment analysis.
2. BERT Model Configuration: The pre-trained BERT model ('bert-base-uncased') and its corresponding tokenizer are loaded using the transformers library. This model is well-suited for NLP tasks and has been fine-tuned for sentiment analysis. The number of labels is set to 3 to accommodate the sentiment classes: negative, positive, and neutral.
3. **Dataset Preparation:** A custom dataset class, TweetDataset, is defined to organize the tweets and labels. The dataset is tokenized using the BERT tokenizer, and DataLoader instances are created for both the training and testing sets.
4. Fine-tuning Parameters: Fine-tuning parameters are configured, including the AdamW optimizer with a learning rate of 5e-5 and the number of training epochs (in this case, 1 epoch for demonstration purposes). The model is moved to the GPU if available.
5. Model Training: The model is trained using a loop over the specified number of epochs. Batches are processed using the DataLoader for the training set. The training loop involves tokenizing the input texts, computing the model's output, calculating the loss, backpropagating, and optimizing the model parameters.
6. Model Evaluation: After training, the model is evaluated on the testing set. The evaluation loop is similar to the training loop but without backpropagation. The predicted labels are compared to the ground truth labels, and classification metrics such as accuracy and a detailed classification report are generated.
7. Output and Logging: Time logging is implemented to track the duration of both the training and evaluation phases. The script outputs key information, including the training and prediction times, accuracy, and the detailed classification report.

This implementation provides a comprehensive and modular approach to sentiment analysis using BERT, facilitating experimentation with hyperparameters and model configurations. The logging and output features enhance transparency and allow for detailed analysis of the model's performance. The script is designed for ease of use, making it accessible for participants in the sentiment analysis competition. The final output looks like the following:





The sentiment analysis model achieved an accuracy of 71.62% on the testing set. The classification report reveals balanced precision and recall across sentiment classes, with class 2 (neutral) showing the highest f1-score at 0.72. The overall performance showcases the model's effectiveness in discerning sentiments from Twitter tweets. The prediction process concluded in approximately 3.62 minutes, highlighting efficient model evaluation.

In our initial Project Plan, our intention was to seamlessly integrate this coding competition with LiveDataLab. Unfortunately, due to challenges with staff availability and a lack of setup instructions, we encountered difficulties obtaining LiveDataLab credentials and Microsoft Azure credits necessary for conducting the competitions on that platform. Consequently, we pivoted our approach by developing a sentiment analysis model and showcasing its application on analyzed tweets. Our aim was to demonstrate that this codebase could be extended and enhanced by students in a competition on LiveDataLab, aligning with the format of past Programming Assignments in this class. Despite not achieving direct integration with LiveDataLab, we received feedback from Teaching Assistants (TAs) indicating that proceeding with the project without direct LiveDataLab integration was acceptable.

Had we implemented the competition on LiveDataLab, the leaderboard structure would feature a column for participants' IDs and another for their F1 scores. Participants would be required to submit their test set results in the form of a CSV file to LiveDataLab. The leaderboard rankings would be determined by the accuracy score, with higher scores placing participants at a more advantageous position on the board.

It's worth noting that, for the purposes of development and execution, we utilized Google Colab. While LiveDataLab integration was initially planned, the adaptation to Google Colab provided a robust environment for running the code, fostering collaboration and ease of access for participants in the sentiment analysis competition. This approach allowed for effective model training, evaluation, and potential future integration with LiveDataLab if the setup challenges are addressed.

# Project Setup:

To execute this project, begin by cloning this repository, ensuring it includes the "archive" folder containing the essential "Tweets.csv" dataset required for model training. Subsequently, navigate to the "CS410\_tweet\_classifier\_model.ipynb" file. Before running the notebook, ensure the following dependencies are installed:

* pip install transformers
* pip install torch
* pip install pandas

Once these dependencies are installed, execute the command jupyter notebook CS410\_tweet\_classifier\_model.ipynb in your preferred environment. This will launch the notebook, allowing you to run the file interactively. Feel free to make adjustments to hyperparameters and the model architecture as needed to enhance performance.

|  |  |
| --- | --- |
| Name | Contribution |
| a. Elias Amuneke | Report, Code |
|  |  |
|  |  |