Elevating Twitter Sentiment Analysis: A Progressive Pipeline for Contextualized and Nuanced Twitter Sentiment Analysis using Transformer Models

1st Krishna Prasad Sheshadri
School of Augmented Intelligence
Arizona State University
Tempe, Arizona, USA
kshesha1@asu.edu

2nd Sameeh Gafoor Suhail
School of Augmented Intelligence
Arizona State University
Tempe, Arizona, USA
sgafoors@asu.edu

3nd Shreyas Kulkarni
School of Augmented Intelligence
Arizona State University
Tempe, Arizona, USA
skulka36@asu.edu

***Abstract*—Sentiment analysis of social media posts provides valuable insights but poses challenges due to informal language. We present a pipeline using natural language processing techniques for contextualized tweet sentiment analysis. Our approach employs using various strategies such as modelling the problem as a question answering task and using various Transformer based models such as BERT and RoBERTa. We also introduce a progressive pipeline that performs enchantments to the pre-trained RoBERTa model by exploring various attention mechanisms in order to improve performance.**

I. Introduction

Sentiment analysis of social media platforms like Twitter has become increasingly important for gaining insights across domains. However, accurately analyzing the sentiment of informal, short tweets poses challenges. In this project, we develop a pipeline for contextualized tweet sentiment analysis using state-of-the-art natural language processing (NLP) techniques.

The objectives include developing techniques capable of accurately identifying sentiment-laden text spans, ensuring computational efficiency for large-scale deployments, and progressively augmenting modeling architectures to enhance contextual understanding. The foundations comprise of standard protocols of data sourcing, preprocessing, cleaning and exploratory analysis to construct a robust pipeline.

Subsequently, the process investigates three modeling approaches - a baseline CountVectorizer model that computes subset-specific term weights and Jaccard similarities, a question-answering methodology framing the task for a fine-tuned DistilBERT model, and an incremental enhancement strategy for RoBERTa-based neural architectures. The first two techniques provide standardized benchmarks, while the third approach constitutes the primary contributions through novel attention mechanisms and representational learning components.

Specifically, the RoBERTa enhancement path initiates with a baseline model leveraging the pretrained capabilities for encoding textual semantics. The first augmentation incorporates multi-head self-attention to capture diverse linguistic relationships within the tweets. Next, a dynamic attention mechanism is introduced to adaptively focus on relevant input subsequences. Finally, the proposed Multi-Head Dynamic Attention model consolidates both attention schemes to provide an integrated representational learning framework.

Extensive empirical evaluation using five-fold stratified cross-validation demonstrates superior extraction accuracy for the novel model in terms of Jaccard similarities between predicted and ground-truth text snippets expressing the sentiment. Beyond quantification, qualitative assessments reveal nuanced improvements in handling informal linguistic varieties. The ensemble strategy across trained models for each fold lends additional robustness.

The implementations facilitate reproducible research and future extensions. Through systematic and progressive enhancements to RoBERTa-centered neural architectures via attention mechanisms, the project provides both theoretical and applied advancements to the multifaceted challenge of social media text analysis. The proposed models display promise for real-world sentiment mining systems involving Twitter data across domains.

A. Background, Problem, Importance

Sentiment analysis stands as a pivotal domain within natural language processing, aiming to decipher human emotions and opinions expressed in textual data. Analyzing sentiments from social media, especially tweets, poses a unique challenge due to their concise and informal nature. The brevity and colloquialism inherent in tweets amplify the complexity of sentiment extraction. Understanding and accurately interpreting sentiments in this context hold immense significance in various domains, including marketing, public opinion analysis, and brand sentiment tracking.

B. Existing Literature

Prior research in sentiment analysis has explored numerous methodologies to comprehend sentiment from text. Existing literature encompasses traditional machine learning approaches, such as CountVectorizer and logistic regression-

based models, to more advanced deep learning architectures like BERT and RoBERTa [1]. Notably, recent advancements in transformer-based models have demonstrated superior performance in understanding contextual nuances within text, setting a benchmark for sentiment analysis tasks.

C. System Overview

The proposed system comprises a systematic approach aimed at improving sentiment extraction from tweets. The process initiates with data collection from a widely recognized dataset sourced from Kaggle [8], followed by meticulous data preprocessing and Exploratory Data Analysis (EDA) procedures. Subsequently, the study delves into three distinct modeling techniques: CountVectorizer for word weight calculation, BERT-based Question-Answering models, and an advanced RoBERTa-based architecture with iterative enhancements.

D. Data Collection

The data mining pipeline for sentiment extraction from tweets begins with critical data preprocessing tasks like cleaning, normalization and exploratory analysis to prepare the Twitter data. Subsequently, three modeling approaches are explored - framing the problem as a question answering task for a fine-tuned BERT model to identify sentiment labels and related text, using a baseline CountVectorizer model to calculate word weights and proportional occurrences, and progressively enhancing RoBERTA architectures by augmenting the baseline with multi-head self-attention to capture contextual relationships, adding dynamic attention for adaptable input focus, and proposing a novel multi-head dynamic attention to combine their strengths. All models attempt to predict the selected text span from each tweet that expresses the associated sentiment. Evaluations are performed using Jaccard similarity scores between predicted and ground-truth text snippets to quantify accuracy. The end-to-end process aims to develop an optimal architecture for extracting sentiment signals from informal Twitter data by systematic comparisons across different modeling paradigms.

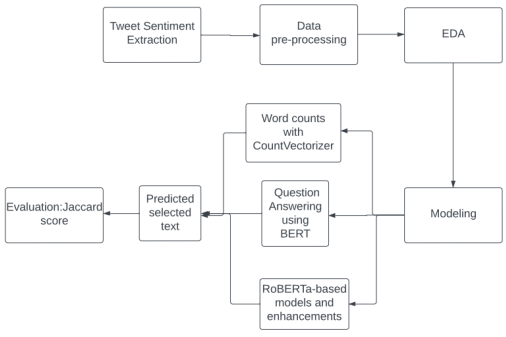

Fig. 1. DM Pipeline.

II. Important Definitions and Problem Statement

A. Transformer

A transformer is a deep learning model architecture that is based entirely on attention mechanisms, without using recursion or convolution. The transformer model consists primarily of two components - an encoder and a decoder.

B. Encoder

The encoder component in a transformer architecture processes the input sequence to generate an internal representational encoding. It captures contextual relationships between the symbols in the input sequence through the use of self-attention and feedforward layers.

C. Decoder

The decoder component in a transformer generates the target sequence, one symbol at a time, from the encoded representation created by the encoder. It uses self-attention over the output symbols and attention over the input sequence from the encoder.

D. CountVectorizer

CountVectorizer is a popular technique in natural language processing used as a baseline method for text classification tasks. It converts text documents into token counts which can be used as features for training machine learning models. Key capabilities include vocabulary indexing, document-term matrix generation, and tf-idf weighting schemes.

E. Standard NLP Downstream Tasks

Some common downstream tasks in natural language processing where pretrained models like BERT and RoBERTa are fine-tuned include:

Sentiment analysis - Categorizing sentiment polarity in text Question answering - Answering questions based on context passages Text classification - Assigning labels to documents based on content Named entity recognition - Identifying entities like persons, locations etc. Text summarization - Generating condensed summaries preserving key ideas

F. Problem Statement

The primary challenge lies in developing models capable of understanding and predicting sentiment-related spans within tweets, encompassing positive, negative, or neutral emotions. This entails handling diverse linguistic styles, informal expressions, and contextual nuances prevalent in social media text.

1) Objectives: Accurate Sentiment Extraction: Develop models capable of accurately identifying sentiment-related spans within tweets, aligning them with their corresponding sentiments (positive, negative, or neutral).

Efficiency and Scalability: Create models that offer high efficiency in sentiment extraction, ensuring scalability to handle large volumes of tweets without compromising performance.

Progressive Model Enhancements: Progressively enhance modeling techniques, moving from foundational methodologies like CountVectorizer to advanced transformer-based architectures like RoBERTa, aiming for continuous improvement in predictive accuracy.

2) Constraints: Computational Resources: Constraints related to computational power and memory allocation may limit the complexity and size of models that can be deployed.

Data Quality and Diversity: The accuracy of sentiment extraction models heavily relies on the quality and diversity of the dataset. Constraints related to biased or limited datasets may impact model performance.

Ethical Considerations: Ensuring ethical handling of data and preserving user privacy remain paramount, constraining the type and extent of data that can be utilized for training sentiment analysis models.

III. Overview of Proposed Approach/System

Our proposed approach consists of several steps ranging from the tweet sentiment extraction to ultimately predicting the selected text. We utilize the popular tweets dataset from Kaggle. Key attributes among the train data include the textID, text, selected text, and the sentiment, where tweets are classified as either positive, negative, or neutral. As part of standard protocols, we perform data pre-processing and concurrently execute EDA procedures where we handle null values, convert text to lowercase, remove punctuation, and eliminate numbers such that our model can produce smoother predictions. In the process, we also create visualizations to help gain a better understanding of the data at hand. Subsequently, the following section explores the three modeling techniques that will assist us in predicting the selected text with high accuracy and efficiency. We investigate two existing modeling techniques and develop a novel, state of the art model that will be further discussed.

A. CountVectorizer

The initial approach is using the CountVectorizer model which is broken down into two crucial steps. First and foremost, it is important to understand the data splitting process that occurs prior to performing necessary calculations. The training data is split into a training set and a validation set, comprising of 80% and 20% respectively, and thus producing subsets of data for their respective sentiments. Once we establish these subsets, we can proceed with the calculations.

1) Word Weight Calculation: Word weights are calculated as the difference between the proportion of tweets containing the word in a sentiment and the sum of the proportions in other sentiments. Therefore, within each of the sentiments, we calculate the weight of the words under the consideration of the proportion of tweets that contain each word. The CountVectorizer model then counts the instances of words in each of the subsets and finds the proportion of tweets that contain each word for each sentiment.

2) Selected Text Calculation: For tweets consisting of positive and negative sentiments, we consider all words within a tweet and calculate the Jaccard score based on the weighted sum of the words. However, for tweets that are of neutral sentiment, we return the entire text. Next, we return the subset with the highest Jaccard score if it exceeds a specified tolerance.

B. Question-Answering Model using BERT

The second approach is performing a modeling technique using BERT as a Question-Answering (QA) problem by leveraging NLP. While formulating the problem, the question considers the sentiment column and whether the sentiment is positive or negative. Furthermore, the context is the text column from the dataset. Once the question is devised, the answer provides us with the selected text column which is used for the prediction.

1) Data Preparation in QA Format: The initial stage of the data preparation process comprises of creating a JSON file for the train and test data. The format of the file is a JSON string that consists of two attributes, the context and the list of questions and answers (QAS). This process serves as the acceptable format in which the `predict()` method can read the dictionaries containing the necessary information.

2) DistilBERT and SQuAD Model: Following the data preparation process, we examine the DistilBERT and SQuAD Model and train `distilbert-base-uncased-distilled-squad` on Kaggle. This NLP model aligns well with the scope of our QA problem and is prevalent within datasets containing English text. Similar to our previous approach, the aim is to produce a Jaccard score that can be used for comparison and evaluate whether it achieves high similarity.

C. RoBERTa based Models

Finally the last approach, that outperforms every other approach previously discussed, is that of using RoBERTa [1]. Our approach involves continually enhancing sentiment analysis models, starting from a stable RoBERTa baseline. Each subsequent step builds upon the previous, introducing novel mechanisms to capture contextual nuances. In the following section, we provide concise yet comprehensive insights into the three pivotal models crafted through this progressive enhancement strategy.

1) Baseline RoBERTa Model: Our starting point is a robust RoBERTa model, a transformer-based architecture pre-trained on vast textual corpora. This model's capacity to capture complex contextual information makes it an excellent starting point for sentiment analysis tasks. The RoBERTa model is fine-tuned for our specific Twitter sentiment analysis dataset.

2) Multi-Head Self-Attention: We augment the model with a multi-head self-attention mechanism to improve its comprehension of contextual relationships within the input sequence. This augmentation entails the creation of multiple attention heads, each of which learns distinct information from the input. These heads run in parallel, enabling the model to

capture diverse dependencies and interactions among words in the sequence. The final representation is a concatenation or weighted sum of the outputs from all attention heads.

3) Dynamic Attention: Building upon the multi-head self-attention model, we introduce dynamic attention to adaptively adjust the attention mechanism. Dynamic attention enables the model to learn attention weights from the input sequence itself, in contrast to static attention weights. We leverage learnable parameters to compute attention scores dynamically, enabling the model to focus on different parts of the input sequence for varying inputs. The goal of this adaptable mechanism is to increase the model's awareness of subtle sentiments and context.

4) Multi-Head Dynamic Attention: In our final enhancement, we propose a novel Multi-Head Dynamic Attention model. Multi-head and dynamic attention mechanisms are combined in this model to maximize their respective strengths. Multiple attention heads capture diverse relationships, while the dynamic attention mechanism adaptively adjusts the focus based on the input context. The model is designed to capture complex sentiment patterns, providing a richer and more expressive representation of sentiment-related information.

IV. Technical details of proposed approaches/systems

In the following section we shall look at the detailed implementation and technical details of the proposed systems.

A. CountVectorizer

The overall architecture of the CountVectorizer model is a continuation of that introduced by Alvi and Talukder [2] where the objective is to transform a subset of text into the form of vectors and in the process, derive their counts. Most techniques during the preprocessing of text, convert the text into a sparse matrix. We shift our focus more towards the calculations of the word weight and selected text such that we can make predictions. This tool utilizes the scikit-learn library in Python.

1) Word Weight Calculation: The algorithm begins by examining the sentiments. Let j represent the class for $j \in \{positive, neutral, negative\}$ and i be all the words in the tweets for each of their respective sentiments. We then calculate $n_{i,j}$ where we find the number of tweets in j containing i and d_j where we simply find the number of tweets in j. At this point, we can compute the proportion of tweets in j that contain i as follows:

$$p_{i,j} = \frac{n_{i,j}}{d_j}$$

The task becomes simpler by creating dictionaries to store words for each sentiment where the values are the proportion of tweets that contain specific words as denoted by $p_{i,j}$. Once this has been calculated, we can calculate the weights assigned to each word within each class $w_{i,j}$ as:

$$w_{i,j} = p_{i,j} - \sum_{k \neq j} p_{i,k}$$

where k represents the iteration over other sentiments in class j. Following this calculation, we see the importance of each word i and can distinguish each of the sentiments in j.

2) Selected Text Calculation: The first step of the algorithm involves determining the sentiment of the tweet, which is denoted by j. If j is neutral, we simply return the entire text as there is no need to execute further extraction of words. Otherwise, with positive and negative sentiments, for all subsets of words in a given tweet, we perform the calculation

$$\sum_i w_{i,j}$$

where i represents the set of words. From this calculation we retrieve the sum of weights for each word. Following this, we return the subset of words with the largest sum under the consideration that it exceeds the particular tolerance, which in the case of our approach, is set to 0.001. The selected text that is returned is classified under the closest possible sentiment. Lastly, the Jaccard score is computed as the similarity metric between predicted selected text and true selected text to quantify how well the model's predictions match the ground truth.

B. Question-Answering Model using BERT

In this model, we use the `distilbert-base-uncased-distilled-squad` derived from the Hugging Face library. San et al. [3] lay the foundation for the various applications for DistilBERT, an improved model. This model can be further fine-tuned to accomplish large-scale tasks. In addition, DistilBERT offers several enhancements from its predecessors such as the ability to reduce size by 40%, retain 90% of its language understanding features, and improve performance by 60%. Fine-tuning the DistilBERT model with the SQuAD model allows for a more QA focus while achieving a significantly improved score. We shall now further investigate the inner workings of the system.

1) DistilBERT and SQuAD Model: After the data preparation process in QA format as a JSON file, we utilize the `QuestionAnsweringModel` from the `simpletransformers` library which uses transformer-based models to achieve the QA problem. We then train the model while taking into account the various parameters of the `train.JSON` file. As part of the QA model parameters, the model is the general-purpose language representation model and the model path represents the path specific to the DistilBERT model. A dictionary further comprises of boolean values for the reprocessing of input data and overwriting of the output directory, learning rate for training the model, number of epochs, maximum length of the input sequences, stride of the input documents' processing capability, and the boolean value for allowing 16-bit floats.

These parameters and their values consist of the following: the model set to `distilbert`, model path set to `distilbert-base-uncased-distilled-squad`, the dictionary with `reprocess_input_data` set to `True`, `overwrite_output_dir` set to `True`, `learning_rate` set to 5×10^{-5}, `num_train_epochs` set to 3, `max_seq_length` set to 192, `doc_stride` set to 64, `fp16` set to `False`, and lastly, the specification for whether to use GPU acceleration. After training the model, we calculate the Jaccard score similar to previous models.

C. RoBERTa based Model

RoBERTa builds upon the transformer architecture, as introduced by Vaswani et al. [4]. Although the transformer architecture consists of an encoder-decoder structure, we concentrate on the encoder in our task of sentiment analysis. Figure 2 represents the Transformer architecture [4].

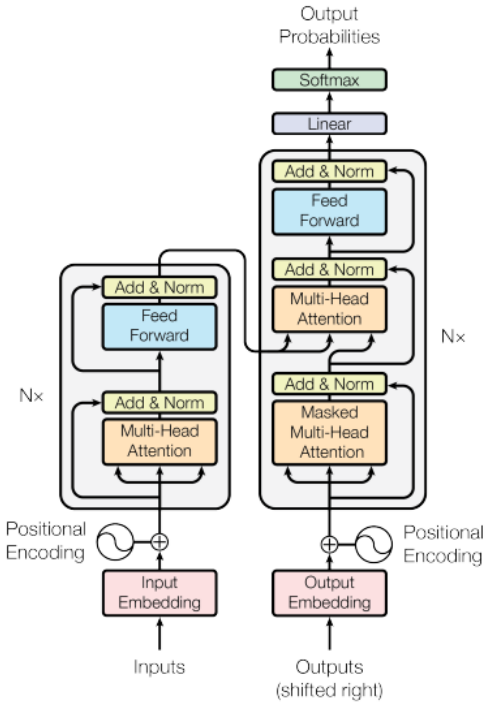

Fig. 2. Transformer architecture.

The encoder is made up of several layers, each of which is divided into two sub-layers: a position-wise fully connected feedforward network and a multi-head self-attention mechanism. Let X represent the input sequence, L be the number of layers, and d_{model} be the dimensionality of the model.

For a given layer l, the output X_l is computed as follows:

$$X_l = \text{LayerNorm}(X_{l-1} + \text{SubLayer}(X_{l-1}))$$

where SubLayer is the combination of multi-head self-attention and feedforward layers. The self-attention mechanism allows the model to weigh different words in the input sequence differently, capturing contextual dependencies.

We shall now discuss further technical details regarding our implementation of the Baseline RoBERTa model. Our implementation utilizes the TensorFlow and Keras frameworks.

The three tensors that make up the model's input are the token types, attention mask, and input sequence, respectively, represented by the tensors `ids`, `att`, and `tok`. The model parameters, including the number of layers, attention heads, and hidden dimensions, are configured by loading the RoBERTaConfig from a pre-specified JSON file.

Next, we use `TFRobertaModel` from the Hugging Face Transformers library to instantiate the RoBERTa model. This model is pre-trained on a large corpus and provides a robust contextualized representation of the input text.

Two parallel branches are created in accordance with the RoBERTa model to predict the beginning and ending locations of the text that was selected. Each branch consists of a dropout layer to mitigate overfitting, a 1D convolutional layer with a kernel size of 1, and a softmax activation function. The model is compiled using categorical cross-entropy loss and the Adam optimizer with a learning rate of 3×10^{-5}.

The baseline model is the basis for further improvements, such as multi-head self-attention and dynamic attention mechanisms, which are covered in the following sections.

D. Multi-Head Self-Attention

We extend the standard RoBERTa model by adding a multi-head self-attention mechanism in an effort to capture richer contextual dependencies. Inspired by the transformer architecture's attention mechanism, this enhancement enables the model to focus on different parts of the input sequence simultaneously.

The self-attention mechanism, or scaled dot-product attention, computes attention scores based on the dot product of the query (Q), key (K), and value (V) matrices. For a given input sequence X, the attention score A is computed as:

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimensionality of the key vectors. Multi-head attention involves creating h different sets of query, key, and value matrices, and the final output is a concatenation of the outputs from all attention heads.

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)$ and W^O is a learnable weight matrix.

We shall now discuss further technical details regarding our implementation of the multi-head self-attention mechanism. The multi-head self-attention is incorporated after the RoBERTa encoder layers. Specifically, it leverages the `MultiHeadAttention` layer of the TensorFlow and Keras libraries. The number of attention heads (*attention_heads*) and the key dimension (*key_dim*) are hyperparameters that can be altered to achieve the desired level of model complexity.

The attention mechanism is applied to the output of the RoBERTa model ($x[0]$) for all three inputs: `ids`, `att`, and `tok`. This multi-head self-attention mechanism generates a set of attention-weighted output representations for each input sequence. The multi-head self-attention outputs are then combined with the original RoBERTa representation using element-wise addition.

The model architecture remains consistent with the baseline after multi-head self-attention is integrated. It includes dropout layers, 1D convolutional layers, and softmax activation functions to predict the beginning and ending positions of the selected text.

E. Dynamic Attention

In order to further improve contextual understanding, we introduce the dynamic attention mechanism as part of our ongoing effort to develop an advanced sentiment analysis model. The dynamic attention model allows the model to dynamically focus on different segments of the sequence by adaptively adjusting attention weights based on the input sequence.

1) Adaptive Attention Scores: Dynamic attention introduces adaptability by computing attention scores dynamically based on the input sequence. The attention scores α_i are calculated using a learnable parameter W_α and a non-linear activation function f:

$$\alpha_i = f(XW_\alpha)$$

2) Dynamic Attention Mechanism: These attention scores are then used in the standard attention mechanism:

$$\text{DynamicAttention}(Q, K, V) = \text{Softmax}(\alpha(QK^T))V$$

We shall now discuss further technical details regarding our implementation of the dynamic attention mechanism. Following a similar implementation strategy as the previous section, the dynamic attention mechanism is implemented as a custom Keras layer, termed `DynamicAttention`. This layer is incorporated into our sentiment analysis model after the RoBERTa encoder layers. Similarly, we also have the hyperparameters *key_dim* and *value_dim*.

The `DynamicAttention` layer processes the input sequence's query, key, and value using dense layers for dimensionality adjustments. The attention mechanism itself is facilitated by the `Attention` layer from TensorFlow, utilizing scaled dot-product attention.

The original RoBERTa representation is then supplemented with the output of the dynamic attention mechanism, which increases the model's sensitivity to changing contextual nuances.

F. Multi-Head Dynamic Attention

The proposed Multi-Head Dynamic Attention model combines the benefits of multi-head attention and dynamic attention. For each attention head, dynamic attention scores are computed independently, providing a diverse set of attention weightings. The outputs from different attention heads are concatenated or averaged to form the final representation.

$$\text{MH}_{\text{Dyn}}(X) = \text{Concat}(\text{DA}_{\text{head}_1}, \ldots, \text{DA}_{\text{head}_h})W^O$$

where

$$\text{DA}_{\text{head}_i} = \text{Att}(XW_i^Q, XW_i^K, XW_i^V)$$

represents the Dynamic Attention for the i-th head, and

$$\text{MH}_{\text{Dyn}}(X)$$

denotes the MultiHeadDynamic mechanism applied to the input X.

We shall now discuss further technical details regarding our implementation of the multi-head dynamic attention mechanism. Once again following a similar structure of implementation as in the previous sections of multi-head self-attention mechanism and the dynamic attention mechanism, a custom Keras layer called `MultiHeadDynamicAttention` is used to implement the Multi-Head Dynamic Attention model. It incorporates multiple attention heads to diversify the focus of the model and operates on the query, key, and value of the input sequence. Once again, we now have the hyperparameters *key_dim*, *value_dim*, and *num_heads*.

The outputs from different attention heads are then either concatenated or averaged. In our implementation, we utilize both approaches and provide the option to choose between them. Concatenation diversifies the information captured by different heads, while averaging ensures a more consolidated representation.

The model's capacity to adaptively focus on different elements of the input sequence is enhanced when the aggregated Multi-Head Dynamic Attention is added to the original RoBERTa representation, once again resembling the above implementations. Once this improvement is made, the model still uses its fundamental components, which are 1D convolutional layers, dropout layers, and softmax activation functions to predict the beginning and ending positions of the text that was selected.

Model Training

We shall now discuss the training of the above models. The training procedure for the sentiment analysis model follows a robust stratified K-fold cross-validation approach, which ensures an exhaustive evaluation of the model's performance across various data subsets. The dataset is divided into five folds, and the model is trained for three epochs on each fold. Within each fold, each of the above discussed RoBERTa-based model architectures are employed.

During training, the model is configured to minimize the categorical cross-entropy loss, a common choice for classification tasks. The training process is supervised by monitoring the validation loss, and the weights of the best-performing model on the validation set are saved. By doing this, overfitting is

less likely to occur and the model will perform well when applied to new data.

An interactive display of the training progress gives information about the convergence and performance of the model on the training and validation sets. Furthermore, for every fold, the Jaccard similarity coefficient is calculated, providing a quantitative measure of the model's ability to accurately predict the sentiment-related spans within the text. The iterative training across multiple folds allows for a comprehensive assessment of the model's robustness and generalization capability. The ensemble of these individually trained models is then leveraged to make predictions on the test set. Note that we use a single P100 GPU for training.

V. Experiments

A. Data description

For our experiments, we utilized the Tweet Sentiment Extraction dataset from Kaggle [8] that contains the following: `train.csv`, `test.csv`, and `sample_submission.csv`.

The datasets consist of the following columns:

- `textID` - a unique identifier for each piece of text.
- `text` - the actual text content of the tweet.
- `sentiment` - the general sentiment associated with the tweet.
- `selected_text` - [available in the training set only] the specific text segment that encapsulates the sentiment of the tweet.

When parsing the CSV files, it is crucial to remove the beginning and ending quotes from the `text` field. This ensures that the quotes are not included in the training process.

Finding the word or phrase from the tweet (`text`) that most accurately captures the given sentiment (`sentiment`) is the main objective of our prediction task. The `selected_text` column in the training set, which offers the ground truth for the sentiment representation, lends additional support to this.

For each of the different approaches implemented, that data was cleaned and pre-processed in a certain manner. We shall now discuss the data pre-processing that was performed for modeling it as a Question-Answering problem and modelling using RoBERTa.

1) Question-Answering Model using BERT: The data preparation process involves creating a JSON file for the train and test data. The format of the file is a JSON string that consists of two attributes, the context and the QAS. The context represents the tweet and its entirety. The QAS is dissected into the numerical ID of the question, the actual question regarding the tweet's sentiment, and the answers consisting of the selected text and the index of where the selected text begins. Below we see an example JSON string from the `train.JSON` file comprising of the context as well as the QAS and its properties in the form of key-value pairs.

```
train_data = [
{
    'context': "This challenge is great",
    'qas': [
        {
            'id': "00001",
            'question': "positive",
            'answers': [
                {
                    'text': "is great",
                    'answer_start': 5
                }
            ]
        }
    ]
}
]
```

2) RoBERTa based Models: We use a methodical preprocessing step in our sentiment analysis task to convert the unprocessed text data into a format that can be used to train our model. For the purpose of later model training, we map sentiment labels ('positive,' 'negative,' and 'neutral') to integer IDs by utilizing the Byte-Level BPE tokenizer from the 'tokenizers' library. Every entry in the training data goes through a sequence of processes that make tokenization and subsequent model input preparation easier. Following text cleaning, the code finds the characters that overlap from the original tweet to the selected text and marks them. The text is tokenized by using the tokenizer, and character offsets are determined for every token in the source text. Special tokens, including [CLS], [SEP], [SEP], a sentiment token, and [SEP], are incorporated at the beginning and end of the tokenized sequence. Upon processing, the data comprises binary arrays that represent the beginning and ending positions of the chosen text, input token IDs, and an attention mask. The preprocessing steps for text data are inspired by the BERT embedding layer. The specific aspects influenced by BERT include tokenization, attention mask creation, and the addition of special tokens. The BERT embedding layer is shown in Figure 3.

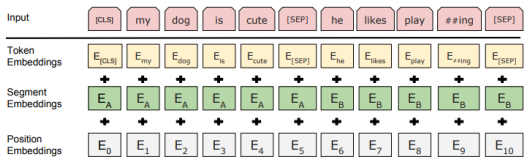

Fig. 3. Embedding Layer of BERT Model.

B. Evaluation metrics

The Jaccard similarity, a commonly used statistic for tasks involving text overlap and segmentation, is used to measure the model's performance. This evaluation metric is particularly relevant to the problem at hand, which entails predicting the word or phrase from a tweet that encapsulates the provided sentiment. The following is the equation for Jaccard Score:

$$\text{Jaccard Score} = \frac{1}{n}\sum_{i=1}^{n}\text{Jaccard}(gt_i, dt_i)$$

Here, n represents the number of documents (tweets), and Jaccard(gt_i, dt_i) is the Jaccard similarity score between the i-th ground truth (gt_i) and the i-th prediction (dt_i). The specific calculation is as follows:

```
def jaccard(str1, str2):
    a = set(str1.lower().split())
    b = set(str2.lower().split())
    if (len(a)==0) & (len(b)==0): return 0.5
    c = a.intersection(b)
    return float(len(c))/(len(a)+len(b)-len(c))
```

C. Baseline Methods for Comparison

To establish a solid baseline for comparison, we select a widely used traditional machine learning technique: the CountVectorizer model. This traditional method computes word weights and Jaccard scores inside sentiment subsets. By considering the frequency and distribution of words, this method serves as a benchmark for traditional feature-based sentiment analysis. We can evaluate the effectiveness of our suggested strategies by comparing them to this baseline method. In the subsequent section, we will analyze the enhancements and novel models, comparing their performance to this baseline to showcase the advancements achieved by the models on Twitter data.

Note that we could have also utilized the RoBERTa model as a baseline; however, we deliberately refrain from doing so to avoid confounding the comparison. Since our primary goal is to showcase the advancements achieved through progressive enhancements, introducing a sophisticated model like RoBERTa as a baseline could overshadow the nuances brought by the proposed mechanisms. By opting for the CountVectorizer model, a simpler yet widely recognized approach, we emphasize a clear contrast between traditional methods and our novel, more intricate models.

D. Results and Overall Performances

In the following section, we present our results and analyze the Jaccard scores for each of the approaches. It is imperative to emphasize that this paper introduces a potential solution to the Kaggle Tweet Sentiment Extraction challenge. Acknowledging the benchmark set by the winning solutions with state-of-the-art performance reaching approximately 0.73 Jaccard scores, our presented results showcase a competitive approach. While our approach may not surpass the current state-of-the-art, it notably converges closely, demonstrating a substantial level of competitiveness.

The CountVectorizer model, serving as the baseline, has the lowest Jaccard score. The low score can be accounted for due to the simplistic nature of the model that may struggle to capture the nuanced patterns within the sentiment data. Notable improvements above the CountVectorizer baseline are demonstrated by the Baseline RoBERTa Model and the QA Model utilizing BERT, both of which are based on powerful pre-trained language models. In the context of the highly competitive nature of the task at hand, it is imperative to underscore that marginal enhancements can yield substantial improvements in Kaggle ranks. And hence we see that minor improvements in the Jaccard score may potentially result in notable advancements in performance, ultimately influencing the final standing on the Kaggle leaderboard. And hence it is important to note the change in Jaccard score in each of the following models.

Models/Approaches	**Jaccard scores**
CountVectorizer (Baseline)	0.65748
QA Model using BERT	0.69451
Baseline RoBERTa Model	0.70454
Multi-Head Self-Attention	0.70433
Dynamic Attention	0.70454
Multi-Head Dynamic Attention (Averaged)	0.70585
Multi-Head Dynamic Attention (Concatenated)	0.70180

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT MODELS/APPROACHES IN TERMS OF JACCARD SCORES.

We see a significant improvement of the Jaccard score,in context of the following task, of Baseline RoBERTa Model as compared to QA Model utilizing BERT. The Multi-Head Self-Attention and Dynamic Attention models, both utilizing attention mechanisms, demonstrate competitive performances. This indicates that attention mechanisms contribute positively to sentiment extraction.

The proposed novel model, the Multi-Head Dynamic Attention model with Averaged Attention, achieves the highest Jaccard score. This implies that the model performs better when its capacity to detect nuanced sentiment patterns is increased by combining several attention heads with dynamic attention methods. The performance of the Multi-Head Dynamic Attention employing Concatenated Attentions is somewhat lower. This indicates that the strategy of concatenating attentions from multiple heads may not be as effective as averaging them. We reason that concatenation may introduce noise or redundancy, leading to a slight decrease in performance. It is also important to note that the following project does not optimize the hyper-parameters using various strategies such as Grid search or Bayesian optimization as this would lead to extremely large training times.

VI. RELATED WORK

Cheruku et al. [5] discusses the role of social media companies, such as Twitter, and how varying decisions and diverse opinions on the platform influence methods that assess the sentiments of customers. Despite NLP being considered the default method in which sentiment analysis is conducted and opinionated data is extracted, previous work has shown that Ontology-based analysis has been primarily used to extract terms and produced scores with relatively high precision and accuracy for text that is emotive in nature. One particular reason for this is that Ontology has successfully transitioned from psychology-based applications to computer science-based applications over the last few decades. The authors'

main objective, however, is to further improve the accuracy of sentiment analysis. Similar to our approach, they achieve this by modifying the RoBERTa model such that only relevant and contextualized information is extracted. The proposed solution of this paper involves combining this approach with Recurrent Neural Networks (RNN) and using the tweets dataset from Kaggle. Along with RNNs, other models were employed such as single-layer and bi-directional LSTMs. The steps of the novel approach consist of first pre-processing the tweets, feeding the modified pre-trained RoBERTa model with the produced tokens to create tensor outputs, using the tensor outputs to fine-tune the model, and lastly passing the selected text to the RNN for appropriate classification. With regards to the results, the proposed RoBERTa-based model of the paper yielded improved accuracy, precision, and recall as well as a higher Jaccard score of 0.733, compared to 0.677 and 0.708 for the BERT-based model and basic RoBERTa model respectively. Likewise, in terms of the RNN architecture, bi-directional LSTMs produced the best possible accuracy of around 84% compared to Single RNN and Single Layer LSTM which were slightly lower. Therefore, the the use of a modified context-enhanced RoBERTa model for target text selection and RNN models for classification provided state-of-the-art accuracy for Twitter sentiment analysis.

Wu et al. [6] explores the concept of Aspect-based Sentiment Analysis (ABSA), which is a subclass of sentiment analysis. Previous methods surrounding ABSA have involved the extraction of high-frequency sentiment polarity words and others have involved using a SentiWordNet scheme to identify semantic pairs in subsets of words. The authors mention that although they seem effective to a certain degree, the underlying problem is that some words do not classify as any of the three sentiments and as a result, relying on this expectation can negatively impact the outcomes. To address this, the paper provides insight into two methods that are helpful in executing ABSA tasks, phrase trees and dependency trees. Phrase trees offer better support for short-distance dependencies between words while dependency trees are better with long-distance dependencies between words. These differences often lead to the prediction of the wrong sentiment among other fallacies. Therefore, the authors propose a novel solution called Phrase Dependency Relational Graph Attention Network (PD-RGAT) that individually extracts and analyzes words in a sentence through the use of directed edge labels and phase information, as well as a Bi-LSTM that retains all representations of words. This is similar to the method in which [5] applied a confusion matrix of Bi-LSTM on full-text data. However, one difference between the two methodologies is that [5] performs their sentiment classification and analysis on only the Twitter dataset while [6] performs theirs on the Twitter, Restaurant, and Laptop datasets. The authors of this paper ultimately aimed to bridge the gap between exploring short and long dependencies between words in a sentence with more ease.

Among the various keyword extraction methods today, Issa et al. [7] focus their work on conducting a comparative study on embedding models using the KeyBERT method. The KeyBERT method is composed of three steps, where the first of which is to select a list of candidate words or phrases. This is done by using the Scikit-Learns library. The next step is the embedding process where embedding occurs between the text and its candidate keywords. Following this, the cosine similarity is calculated between individual keyword vectors and the document vector. While the main focus of the paper is on the second step of the process, the existing problem is that there is a lack of comparative studies that discuss the various models in KeyBERT. Therefore, the authors propose to do a comparison on the performance of the embedding models. For example, when evaluating the performance of the keyword-based `Paraphrase-mpnet-base-v2` model, similar to the Jaccard score, the authors establish metrics such as f1_score, precision, recall and MAP. Following their analysis, they come to the conclusion that the model performs worse with longer datasets containing texts and similar observations are made for other tested models such as the `Paraphrase-distilroberta-base-v2` and even keyphrase-based `Longformer` model. From this paper, we can draw parallels to the CountVectorizer model where either the simple nature of the model or the large amounts of sentiment data, that may consist of nuanced patterns, account for lower scores for a particular evaluated metric. There exist various tradeoffs between models such as longer processing time or inability to handle large datasets, however through this comparitive study, the authors pave the way for future researchers to make the necessary improvements to achieve better performance and overall efficiency.

VII. Conclusions

In this paper, we proposed a pipeline to solve the Kaggle challenge of Tweet Sentiment Extraction using natural language processing techniques. Our approaches focused on leveraging state-of-the-art models like BERT and RoBERTa for the task of sentiment extraction in tweets. Our findings reveal that the CountVectorizer model, serving as the baseline, exhibited the lowest Jaccard score. This is attributed to its simplistic nature, which may struggle to capture nuanced patterns within sentiment data. Notable improvements were observed with the Baseline RoBERTa Model and the QA Model using BERT, both leveraging powerful pre-trained language models.

The highly competitive nature of the task underscores the importance of marginal enhancements. We propose a novel Multi-Head Dynamic Attention RoBERTa model, which when used with averaging attentions, proves to be the best performing model in our study. This demonstrates the efficacy of combining multi-head dynamic attention to capture diverse linguistic relationships and dynamic attention to focus on salient parts of the input text.

In conclusion, our study contributes to the understanding of sentiment extraction in tweets by providing insights into the performance of various models and the importance of attention mechanisms. Our primary contribution is the novel mechanism of Multi-Head Dynamic Attention, which proves to have great potential. Future work would involve explore

hyperparameter optimization methods as well as strategies to reduce the large training times. Subsequent research endeavours may investigate supplementary model improvements, dataset modifications, or fine-tuning techniques to further enhance performance in this challenging domain.

REFERENCES

[1] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," arXiv preprint arXiv:1907.11692, 2019.

[2] N. Alvi and K. H. Talukder, "Sentiment Analysis of Bengali Text using CountVectorizer with Logistic Regression," in 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 01-05.

[3] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," in arXiv preprint arXiv:1910.01108, 2019.

[4] A. Vaswani et al., "Attention is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), Long Beach, USA, 2017, vol. 30.

[5] R. Cheruku, K. Hussain, I. Kavati, A. M. Reddy, and K. S. Reddy, "Sentiment classification with modified RoBERTa and recurrent neural networks," in Multimedia Tools and Applications, Hyderabad, India, 2023

[6] H. Wu, Z. Zhang, S. Shi, Q. Wu, and H. Song, (2022, January) "Phrase dependency relational graph attention network for aspect-based sentiment analysis". *Knowledge-Based Systems*[Online]. vol. 236, issue: 107736. Available: https://www.sciencedirect.com/science/article/pii/S0950705121009734?casa_token=jN4lXBl3_A4AAAAA:BNBxGTwWrZ-lkYZFNJK8kamaLa5iiM36aGTQX-_tP9rXACY6deWEblGXG3rVNFpl_PRNlW4n45E

[7] B. Issa, M. B. Jasser, H. N. Chua and M. Hamzah, "A Comparative Study on Embedding Models for Keyword Extraction Using KeyBERT Method," 2023 IEEE 13th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 2023, pp. 40-45, doi: 10.1109/ICSET59111.2023.10295108.

[8] https://www.kaggle.com/competitions/tweet-sentiment-extraction/overview

The code for this paper is available on GitHub at: https://github.com/kshesha1/CSE-572-Project