BRAIN AND COGNITIVE SOCIETY

TWITTER SENTIMENT ANALYSIS

TSE GROUP-3

TEAM MEMBERS:

- Atul
- Sajal Goyal
- Saurabh Gupta
- Saad Ahmad

Abstract[1]:-

With all of the tweets circulating every second it is hard to tell whether the sentiment behind a specific tweet will impact a company, or a person's brand for being viral (positive), or devastate profit because it strikes a negative tone. Capturing sentiment in the language is important in these times where decisions and reactions are created and updated in seconds. But, which words actually lead to the sentiment description? In this competition, you will need to pick out the part of the tweet (word or phrase) that reflects the sentiment.

Help build your skills in this important area with this broad dataset of tweets. Work on your technique to grab a top spot in this competition. What words in tweets support a positive, negative, or neutral sentiment? How can you help make that determination using machine learning tools?

In this competition, we've extracted support phrases from Figure Eight's Data for Everyone platform. The dataset is titled Sentiment Analysis: Emotion in Text tweets with existing sentiment labels, used here under creative commons attribution 4.0. International license. The objective in this competition was to construct a model that can do the same - look at the labeled sentiment for a given tweet and figure out what word or phrase best supports it.

Introduction:-

We are given train.csv, test.csv, and sample_submission.csv.

Type of Data:

Each row contains the text of a **tweet** and a **sentiment label**. In the training set we are provided with a word or phrase drawn from the tweet (**selected_text**) that encapsulates the provided sentiment. When parsing the CSV, to remove the beginning / ending quotes from the text field, we don't include them in our training. We're attempting to predict the word or phrase from the tweet that exemplifies the provided sentiment. The word or phrase should include all characters within that span (i.e. including commas, spaces, etc.).

[1] Code Available here - https://github.com/sajalgoyal113/nlp_tse

The format is as follows:
<id>,"<word or phrase that supports the sentiment>"
For example:
2, "very good"
5, "I am neutral about this"
6, "bad"
8, "if you say so!"

Files:

- train.csv - the training set
- test.csv - the test set
- sample_submission.csv - a sample submission file in the correct format

Columns:

- textID - a unique ID for each piece of text
- text - the text of the tweet
- sentiment - the general sentiment of the tweet
- selected_text - [train only] the text that supports the tweet's sentiment

Motivation:

There's a fast-growing collection of useful applications derived from this field of study. They range from simple to complex. Below are a few of them:

- ❏ Spell Checking, Keyword Search, Finding Synonyms.
- ❏ Extracting information from websites such as: product price, dates, location, people, or company names.
- ❏ Classifying: reading level of school texts, positive/negative sentiment of longer documents.
- ❏ Machine Translation.
- ❏ Spoken Dialog Systems.
- ❏ Complex Question Answering.

Indeed, these applications have been used abundantly in industry: from search (written and spoken) to online advertisement matching; from automated/assisted translation to sentiment analysis for marketing or finance/trading; and from speech recognition to chatbots/dialog agents (automating customer support, controlling devices, ordering goods).

Theory-:

Everything we express (either verbally or in written) carries huge amounts of information. The topic we choose, our tone, our selection of words, everything adds some type of information that can be interpreted and value extracted from it. In theory, we can understand and even predict human behaviour using that information. But there is a problem: one person may generate hundreds or thousands of words in a declaration, each sentence with its corresponding complexity. If you want to scale and analyze several hundreds, thousands or millions of people or declarations in a given geography, then the situation is unmanageable. Data generated from conversations, declarations or even tweets are examples of unstructured data. Unstructured data doesn't fit neatly into the traditional row and column structure of relational databases, and represent the vast majority of data available in the actual world. It is messy and hard to manipulate. Nevertheless, thanks to the advances in disciplines like machine learning a big revolution is going on regarding this topic. Nowadays it is no longer about trying to interpret a text or speech based on its keywords (the old fashioned mechanical way), but about understanding the meaning behind those words (the cognitive way). This way it is possible to detect figures of speech like irony, or even perform sentiment analysis.

The best way to understand any dataset is by doing exploratory data analysis (EDA). EDA summarizes the main characteristics, often with visual methods. EDA is an important step before diving into applying machine learning models, It basically refers to the critical process of performing initial investigations on data so as

- ➢ to discover patterns,
- ➢ to spot anomalies,
- ➢ to test hypothesis and
- ➢ to check assumptions with the help of summary statistics and graphical representations.

Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with text data. This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens. BERT is a bidirectional model that is based on the transformer architecture, it replaces the sequential nature of RNN (LSTM & GRU) with a much faster Attention-based approach.

Method:-

Tokenizer used - ***ByteLevelBPETokenizer*** by Hugging Face
Transformer used - ***RoBERTa*** pre-trained model
The pre-processing steps are as follows -

- From the input file, first rows with null values are removed. Now we want to form an input to RoBERTa model and hence we define three variables namely **input_ids**, **attention_mask** and **tokens.** Combination of tweet and sentiment comprises the **input_ids**. **attention_mask** is an array of 1's and 0's in which indices representing the words in **input_ids** are marked 1 while the rest are marked 0. This later helps us in padding. **tokens** represents the word indices in the tweet which should be selected text.
- The above three inputs are fed into the RoBERTa model. So here we are using **RoBERTa** transformer with some additional layers stacked to use it for our purpose. Layers include Conv1D layer, Dense layer, GRU layer, Dropout and Activation functions.
- By stacking layers on the RoBERTa model, we get two outputs: **start_tokens** and **end_tokens**. **start_tokens** gives the index of starting word of selected text in the tweet while **end_tokens** gives the index of last word of the selected text.
- Before training our model, we splitted our training dataset into 5 fold meaning 80% of the data for training and 20% for validation. We train the model with batches of training sets and a particular number of epochs. Number of epochs are selected such that the model does not overfit on the training set.
- Using 5-fold splitting, we get 5 outputs corresponding to each fold of which we take average to reduce overfitting.

Experiments:

- ➢ We tried to change EPOCH, BATCH_SIZE, PAD_ID, SEED, LABEL_SMOOTHING, our optimum results were for :
  - ★ EPOCH =3
  - ★ BATCH_SIZE = 32
  - ★ PAD_ID = 1
  - ★ SEED = 8192
  - ★ LABEL_SMOOTHING = 0.1

- ➢ We tried different combinations of Dropout , Conv1D , LeakyReLU , GRU , Dense , Flatten layers.
- ➢ Activation used : **SoftMax**

- ➢ Then coming to the optimizers and learning rate . We used different optimizers and learning_rate. Our optimum results were for **ADAM** optimizer and learning_rate=**3e-5**.
- ➢ Then we changed the n_splits in StratifiedKFold but with every shuffle we got different results .
- ➢ Later we also used distilbert model (from simpletransformer library) which was already trained on SQUAD. So obviously most of the things were fixed in there already so we couldn't change much. Results of both the models are mentioned below.

Results:

Using Roberta Model our highest score was:

Private Score : 0.71702
Public Score : 0.71569

Using Distilbert(already trained on squad) [distilbert - base- uncased - distilled - squad] our highest score was:

Private Score : 0.69949
Public Score : 0.69904