Project: Tweet Sentiment Extraction

Atul Yaduvanshi, Saad Ahmad, Sajal Goyal, Saurabh K. Gupta

Under Brain and Cognitive Society, IIT Kanpur

Science & Technology Council IIT Kanpur Summer Camp 2020

Abstract

This document is a part of report done under Science and Technology Council (SnT) IIT Kanpur Summer Camp 2020. During the Summer Camp 2020, team participated in Kaggle competition 'Tweet Sentiment Extraction' under Brain and Cognitive Society (BCS) IIT Kanpur. The whole project is guided by Ishika Singh.

In order to keep cognition models accessible, we need it to understand human language. The Natural Language Processing (NLP) become important which is concerned with the interactions between computers and human (natural) languages. One of basic the basic attribute of human communication is Sentiment. It is important that machines understand these sentiments.

Keywords: CNN, LM, NLP, pyTorch, roBERTa, Tensorflow,

Project: Tweet Sentiment Extraction

Plan of Action:

1. Understanding the environment of Kaggle competitions
2. Getting started with Technical definitions
3. Exploratory data analysis (EDA)
4. Applying models and fine tuning
5. Future Work

1. Understanding the environment of Kaggle competitions

Team first familiarizes with the Kaggle platform. How to use the platform, how to find datasets, how to upload datasets, how to download files from Kaggle notebooks. It went through the rules and regulation and registered on the 'Tweet Sentiment Extraction' . The followings are the important deadlines for this competition.

- The maximum team size is 5.
- Submission Limits: Team may submit a maximum of 5 entries per day
- Team may select up to 2 final submissions for judging
- Competition Timeline
    - Start Date: March 23, 2020
    - Team Merger Deadline: May 26, 2020
    - Entry Deadline: May 26, 2020
    - End Date (Final Submission Deadline): June 2, 2020 11:59 PM UTC

2. Getting started with Technical definitions

Team went through various online sources to understand the theory behind like NLP, Word Embedding, Stop words, nGrams, Jaccard Score, Tokenization, Stemming, Lemmatization, Tensor Flow, Convolutional Neural Networks (CNN), etc. Following are the website link of the resources:

- Your Guide to Natural Language Processing (NLP)
- The 7 NLP Techniques That Will Change How You Communicate in the Future (Part I)
- A Practitioner's Guide to Natural Language Processing (Part I) — Processing & Understanding Text
- Natural Language Processing is Fun! - Adam Geitgey
- Word Embeddings for NLP
- Beyond Word Embeddings Part 2
- Intuitive Guide to Understanding GloVe Embeddings
- https://github.com/dipanjanS/practical-machine-learning-with-python/tree/master/bonus%20content/nlp%20proven%20approach
- https://github.com/joshzwiebel/Tweet-Sentiment-Extraction/blob/master/tweet_sentiment_extraction/Exploration/exploration.ipynb
- BERT Explained: State of the art language model for NLP
- https://arxiv.org/pdf/1810.04805.pdf
- https://www.kaggle.com/mohannksr/tensorflow-roberta-cnn-head-lb-v2
- Introduction to sentiment analysis: What is sentiment analysis? |

- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
- Regex Cheat Sheet
- Intuitive Guide to Understanding GloVe Embeddings
- Tutorial for tensor flow.

Team has understood concepts form the above resources. *Basic understanding of various keywords can be included in required.*

3. Exploratory data analysis (EDA)

EDA is an approach to analyzing data sets to summarize their 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.

Team referred various public Kaggle Notebooks under the ‘Tweet Sentiment Extraction’ competition. We understood the EDA of the given dataset, mostly focusing on the features *text, selected_text, sentiment* of train dataset and *text, selected_text* on test dataset. We also understood the python codes behind those EDA. Some of the highlights after EDA:

- We have decided to remove the only one row with null.

- URLs in the *selected_text* is mostly of the *netural* sentiment
- *selected_text* with one only star (*) has *negative* sentiment, replacing all single * with the word with a negative sentiment
- Most of the negative and positive sentiment has word length of 5, netural sentiment are widely spread.
- Dataset is randomly shuffled, with equal percentage of for positive, negative, neutral – approx (31%,28%,49%) in the train and test dataset.

We have understood that the given data is clean and can be directly used in the model.

4. Applying models and fine tuning

As of now we have started on kaggle coding platform to load the data and train our model. Tokenization is considered as a first step for stemming and lemmatization (the next stage in text pre-processing). 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. We are using ByteLevelBPETokenizer by Hugging Face. 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. BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. So here we are using RoBERTa transformer with some additional layers stacked to use it for our purpose. We have achieved a score of 0.708, 0.731 being the highest on the leader board.

5. Future Work

- Use Lemmatization.
- Tweak parameters of the layers to get better accuracy.
- Introduce some punctuation patterns to detect selected text from sentiment.
- Use more stopwords only for positive and negative sentiment sentences.
- We are thinking to explore PyTorch.