Tweet Sentiment Extraction

Yatish Goel · Varenya Srivatava · Tushar Singla · Nishima Panwar · Gagan Aryan

Mentor - Ishika Singh

Abstract

Twitter according to Wikipedia is a micro-blogging and social networking site where users post and interact with messages known as âtweetsâ. These tweets play an integral part in conveying many important messages. A new policy by the government, the retirement of a sportsman, key updates of tech-giants and many other updates are announced first on twitter. Over a billion tweets are tweeted every day. With all these tweets getting circulated every second, it is hard to tell whether the sentiment behind a specific tweet will impact a company, or a personâs, brand for being viral(positive), or devastate profits because it strikes a negative tone. Hence, capturing the sentiment of a tweet is critical. In this project, we not only try to capture a sentiments tweet but also try to extract the phrase that conveys a particular sentiment. To achieve this we have used two NLP techniques NER and Roberta trained models by modelling the problem statement in the form of a question answering task. We obtained an accuracy of 66.4 per cent using a baseline NER model and an accuracy of 70.3 per cent using the Roberta model. This task was also a Kaggle competition that concluded recently. We finished 1313 in the competition out of 2227 team with an accuracy of 71.25 per cent.

I. Introduction

The project started with us learning the various nuances of NLP. We started with the lectures of CS224 learning about the word2vecs(also known as the Skip-Gram model), GloVe Vectors for word representation, dependency parsing(which later helped us to come up with an ingenious solution.

Once a basic of NLP tasks was achieved we went on to have a practical approach to the task. Hence we learnt the preprocessing part using SpaCy and NLTK. The learning involved going through some courses(thanks to free Coursera :P) were we learnt tokenizing and cleaning the dataset.

Now we had to decide on how we would approach the problem. The solution used by others participating in the competition was either using a Named Entity Recognition approach or approaching it as a Question-Answering task using various pre-trained model. We first used the NER approach as it was much obvious and was far easier than the rest. We later shifted to a transformer-based Roberta model to obtain a better accuracy and much better scope of improvement. We had to weigh the pros and cons of using one model over the other so we read the original paper and some comparitive studies of BERT, RoBERTa and ALBERT. It was only after that, that we shifted to RoBERTa approach.

Some analysis was later done on the shortcomings in our solution the data not performing well was manually looked at(for similarity score < 0.1).

II. Competition Metric

The metric in this competition is the word-level Jaccard score. Jaccard similarity or intersection over union is defined as size of intersection divided by size of union of two sets. Let us take example of two sentences:

Sentence 1: AI is our friend and it has been friendly

Sentence 2: AI and humans have always been friendly

In order to calculate similarity using Jaccard similarity, we will first perform lemmatization to reduce words to the same root word. In our case, âfriendâ and âfriendlyâ will both become âfriendâ, âhasâ and âhaveâ will both become âhasâ.

III. Approaches Adopted

Before getting started with modelling our solution we made it a point to look at various Exploratory Data Analysis to have a better understanding of the data we have been given with. For this, we looked at various notebooks that had already done the same(owing to our inexperience to do on our own). So the first approach we adopted was NER.

III.I. Named Entity Recognition

Named Entity Recognition (NER) is a standard NLP problem which involves spotting named entities (people, places, organizations etc.) from a chunk of text, and

classifying them into a predefined set of categories. We used spacy for creating our own customised NER model or models (seperate for each Sentiment).

In our case, NER is used to filter out the string that does not contain the word or phrase that reflects the polarity of the tweet. After this the type of tweet is extracted and mapped to the list of entity type. We have observed that single model is not quite effective. So, we have used separate models to each of the sentiment labels. We used as text as selected text as selected text for all neutral tweets due to their high jaccard similarity. We used selected text for all tweets having number of words less than 3 in text. We trained two different models for both positive and negative tweets. Data has not been preprocessed as the selected text contains raw text. Using this approach we were able to achieve an accuracy of 66.4 percent.

III.II. Q/A using RoBERTa

The current version of the notebook makes use of the bert-base-uncased model. As Transfer Learning from large-scale pre-trained models becomes more prevalent in Natural Language Processing (NLP), operating these large models in on-the-edge and/or under constrained computational training or inference budgets remains challenging. A Robustly optimized method for pre-training NLP systems that improves on Bidirectional Encoder Representation from transformers, or BERT. roBERTa, is a retraining of BERT with improved training methodology, 1000 percent more data and compute power.

We had to convert training data along with test data into the arrays that roBERTa understands. TF 2.0 models accepts only two formats as inputs. We gather all the input as a list of padded length with several input Tensors in the order give in the docstring. Tokens are input into bert model and we use BERT's first output, these are embedding of all input tokens .Now, we apply a one dimensional convolutional layer and transform the embeddings into shape batchsize*MAXLEN,1. We then flatten this and apply softmax. Outputs - x1 are 2 dimensional array of shape batchsize*MAXLEN and all these are hot encoding of the start tokens indices. x2 are the end tokens indices. We train with 5 folds Stratified KFolds (based on sentiment stratification) ,Each fold, the best model weights are saved and then reloaded before validation set prediction and test prediction.

In this work, we propose a method to pre-train a smaller general-purpose language representation model, called Bert Base Uncased, which can then be fine-tuned with good performances on a wide range of tasks like its larger counterparts. This approach gave us an accuracy of 70.3 percent.

IV. Complete overview of our final notebook

We start preparing the data using Bert Tokenizer.

Then the bert pretrained tokenized (uncased) is loaded and saved.

After, we reload and use BertWordPieceTokenizer.

The comment text is prepared and encoded using this tokenizer easily. We set the maxlen=128

We load the pretrained bert ('uncased') transformer layer, used for creating the representations and training our corpus. We then create the representation for the selected text from tweet text . This representation is created such that the positions of tokens which is selected from text is represented with 1 and others with 0.

We then create a multi-input model (comment + sentiment label). Finally, we train the model, output predictions and re-alingn those predictions using tokens.

V. Post Model Analysis

We tried to figure out what might be the possible sentences for which the model seemed to fail. Sarcastic sentences initially seemed one such loophole. After looking at the sentences where the jaccard value was less than 0.1 we came to know that this was not the case on the contrary the test cases for which the model seemed to fail were quite trivial.
Example:
Tweet: Cool!!! No wonder you didn't sleep much
Selected Text: Cool!

Selected by Model: Cool!!!
Tweet: So glad sam is in good mood.
Selected Text: good
Selected by Model: glad

As it is evident from the examples the text selected by the model was not wrong but just different from the actual one. Also, some selected texts were noisy. These tweets that scored less than 0.1 were 9 percent of the total number of tweets in the dataset.

VI. Comparision with the most accurate model in the competition

The main difference between our model and the one that ended up in the top of the leaderboaard was their custom loss function that was based on the jaccard similarity of

various sentences. Our model used the conventional Categorical Cross-entropy. Besides the embedding of the top model also was a lot more rich than any other models used in the competition. The solution also dealt with the cases of low jaccard values separately. Some techniques(that were unheard of for us before) like multi sample dropout, AdamW with linear warm up schedule were also used.

Link to our final implementation

References

[1] RoBERTa - A robustly optimized BERT training approach
[2] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
[3] CS224N Natural Language Processing with Deep Learning
[4] Tweet Sentiment Extraction Analysis, EDA and Model, Notebook on Kaggle
[5] NER trainging using SpaCy Ensemble
[6] Training Model using SpaCy
[7] BERT, ROBERTa, DistillNET which one to use
[8] RoBERTa: An optimized method for pretraining self-supervised NLP systems
[9] Hugging Face Documentation
[10] NLP in Tensorflow by Larence Moroney