PROJECT ON

SENTIMENTAL ANALYSIS

ON SST-5 DATA SET

-AVINASH BABU DEBBATI

ABSTRACT:

Sentiment classification is an important process in understanding people's perception towards a product, service, or topic. I used a promising deep learning model called BERT to solve the fine-grained sentiment classification task. Here, Along with BERT, other NLP models were also used on sst-5 and sst-2 dataset for classification but BERT gives the best accuracy among them.
Index Terms—sentiment classification, natural language processing, language model, pre-training

Introduction:

Sentiment classification is used to classify the given statement into any of the given sentiment classes. Fine grained sentiment analysis has got 5 classes which are very negative, negative, neutral, positive and very positive which are denoted by 0,1,2,3,4 respectively. Whereas in binary classification there are two classes either positive or negative denoted by 0 and 1 respectively. First the main task is to convert the given texts into meaning full vectors which can be processed to our machine learning model, which can be done by many NLP models.
In this project BERT(Bidirectional Encoder Representations from Transformers) and few other
NLP models are used to classify SST(Stanford Sentiment Treebank) dataset.

Dataset:

SST(Stanford Sentiment Treebank) is one of the most used and available datasets which is used for fine grained sentiment classification. I have taken this SST dataset from pytree bank python package which has movie reviews given by various users. It has a training set of 8544 sentences and a testing set of 2210 sentences with corresponding sentiment value.
It has 5 classes: 0-very negative, 1-negative, 2-neutral, 3-positive, 4-very positive. It can be converted into a binary dataset(SST-2) indicating either positive or negative. The data is stored in the form of trees.

Methodology:

The most important task in sentiment analysis is converting a text into a fixed size vector known as embeddings. There are various methods for getting embeddings.

BERT:-

In this project I have used a pre-trained BERT model which basically gives a sentence vector. To get the sentence embedding I have used the sentence transformer package. In bert the transformer encoder directly reads the entire sequence of words at once. BERT uses mainly two training strategies:

a) Masked LM: which is acronymed as MLM.It basically replaces 15% of tokens with [MASK] token and model then predicts original value of the masked words with the help of non-masked words based upon the context.

b) Next Sentence Prediction: Which is also called NSP. It basically gets the pair of sentences and then tries predicting whether they are subsequent to each other or not. 50% of the training data is joined with their succeeding sentence and rest 50% with random sentences from the corpus and then the model tries to predict the next statement.

There are two variants in BERT models one is BERT large and BERT base.

	Bert base	Bert large
Number of layers	**12**	**24**
Number of hidden units	**768**	**1024**
Number of self-attention heads	**12**	**16**
Total trainable parameters	**110M**	**340M**

While taking the input, bert requires the sentences to be tokenized with each sentence starting with [CLS](classifying token) and ending with [SEP](separating token).
For preprocessing the data, convert the sentences into lower case and remove any punctuation marks if there and then tokenize them. All this is done by the Sentence Transformer (‘bert-base-nli-mean-tokens’) command imported from the sentence transformer package.
Now these sentence embeddings of BERT are sent to the neural network(imported from
keras which has input layer with 768 nodes, 2 hidden layers, first layer has 128 nodes and then next layer has 16 nodes, with relu as activation function and adam as optimizer) and then the end layer that is soft max layer predicts the probability of different classes. The class with greatest probability will be the predicted output.

Glove, word2vec etc gives word embeddings instead of sentence embeddings.

GLOVE:-

I have used LSTM(and also Bi-LSTM), taken word embeddings and learned the relations among the word sequences, which is an RNN(helps in classifying, processing and making predictions using the time series data). I have used a pre-trained glove.42B.300D (Common crawl,42B tokens,1.9M vocab,300d vectors) model which has word embeddings of various words from the dataset provided by common crawl. After pre-processing, the tokens are sent to the embedding layer and LSTM will combine these word embeddings and interpret the relation among them. Embedding layer can be used in two ways:

1. Supplying the embedding matrix which consists of word embeddings (taken from GLOVE) of the words from vocabulary. These are used for getting the embeddings of input sequences.
2. In this case it generates its own word embeddings of input sequences. Next is a dense layer and then the softmax layer which predicts the class label. I have used a dropout of 0.2 for avoiding over fitting of the model.

OTHER MODELS: -

I used ‘Textblob’ and ‘Vader lexicon’ pre-trained models which have sentiments associated with particular sentences.

I have also used Pytorch for generating sentence embedding. It basically considers the

number of layers, number of hidden units(of BERT model) and sentence id. According to the results given by the experts the more accurate embedding is produced by concatenating the output of the last 4 layers(among the 12 layers of BERTbase model).

I have also checked the accuracy for few other machine learning classifiers like logistic

regression, k-nearest neighbors, gradient boost etc.

EXPERIMENTS AND RESULTS:

METHOD	ACCURACY ON SST2 DATA SET	ACCURACY ON SST5 DATA SET
LSTM(WITH GLOVE EMBEDDINGS)	76.83%	33.89%
Bi-LSTM(WITH GLOVE EMBEDDINGS)	77.32%	32.126%
LSTM (WITH RANDOM EMBEDDINGS)	76.108%	38.95%
BI-LSTM (WITH RANDOM EMBEDDINGS)	74.25%	38.099%
BERT-NN	84.16%	**49.29%**
BERT-LOGISTIC REGRESSION	**84.9%**	46.87%
Bert-random forrest	83.03%	46.96%
Bert _KNN	80.31%	42.21%
BERT-GB	84.02%	48.28%
BERT_GNB	79.77%	41.90%
BERT-XGB	84.30%	48.19%
TEXT BLOB	20.45%	28.37%
VADER LEXICON	20.95%	31.53%
Bert -SMX	80.05%	41.17%

Note :-

BERT-Bi Directional encoder representations from transformer
LSTM-long short term memory
BiLSTM- Bi directional Long Short Term Memory

NN-Neural Network

LR-Logistic regression

KNN-KNeighboursclassifier

rf-RandomForestclassifier

gb-GradientBoostingClassifier

gnb-GaussianNB

xgb-XGBClassifier

smx-SGDCclassifier