|  |
| --- |
| **Text Classification on News Articles using BERT Framework** |
|  |
| *College of Computing and Informatics  Drexel University*  *Philadelphia, PA*  **Team**  Madhu Bandru ([mb4236@drexel.edu](mailto:mb4236@drexel.edu))  Spandana Bendi ([sb4262@drexel.edu](mailto:sb4262@drexel.edu))  Vuthej Krishna Reddy Verama Reddy (vv334@drexel.edu) |
|  |
|  |
|  |
|  |

Abstract

|  |
| --- |
| 050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099 |

|  |
| --- |
| 000  001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049 |

Text classification is a traditional NLP task of assigning predefined class for the text. In this paper, we will be discussing about implementing a technique for fine-tuning a language model that will be further applied on Text Classification task. Precisely, we will be working on categorizing a news article into one of the four different classes (World, Sports, Business, Sci/Tech) based on the article's text. Instead of training the model from scratch, we will be fine-tuning a pre-trained model, which is also known as transfer learning. Bidirectional Encoder Representations from Transformers (BERT) is a transformers-based machine learning technique specifically aimed at Natural Language Processing tasks which is also the framework that we will be utilizing in this term project. Because of its bidirectionality, it utilizes all of the text from the document before arriving at the meaning of each word.

Motivation

Natural Language Processing (NLP) has many applications such as Sentiment Analysis, Text Classifications, Chatbots & Virtual Assistants, Text extractions, Text summarization and many more. Our team explored and worked on projects related to text extraction, sentiment analysis and text summarization individually. Now, for this term project we are interested to work on “Intent Classification” task. Intent Classification is a classification problem, which assigns a label to the given user query. This will be a multiclass-classification where the user queries will be labelled from four different categories. For example query “**Reuters - Private investment firm Carlyle Group,\which has a reputation for making well-timed and occasionally\controversial plays in the defense industry, has quietly placed\its bets on another part of the market**.” is labelled as “**Business**” while query “**AP - Kevin Hartman made seven saves for Los Angeles, and Jon Busch had two saves for Columbus as the Galaxy and Crew played to a 0-0 tie Saturday night.**” is labelled as “**World**”. These examples are samples from “AG’s News Corpus Data”.

Intent classification plays a crucial role in business growth. Intent classifier identifies the actual intent of the customer and helps in suggesting a valuable/accurate service. Everyday there will be thousands of customer queries in different firms or applications. Manual attention for every query is a hectic task and this is where the intent classifier makes this job easy.

Data

For this project we are going to use "[AG's Corpus of New Articles](http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html)" [4]. AG is a collection of more than 1 million news articles which have been gathered from more than 2000 news sources by ComeToMyHead [4] in more than 1 year of activity. ComeToMyHead [4] is an academic news search engine which has been running since July 2004.

This dataset comprises of both train and test data. Train set consists of 4 different classes and 30,000 instances for each class which totals to 1,20,000 instances wherein test set is built of 1,900 instances for each class which totals to 7,600 instances across the four classes.

|  |
| --- |
| 100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149 |

Current dataset is made up of three columns, namely,

* Class
* Title
* Article

‘Class’ attribute is our target variable with four different classes while ‘Article’ attribute is our focused text data which has the text of complete news article.

Text data from ‘Articles’ attribute is then pre-processed to remove the presence of special characters, delimiters and extra spaces further to which the sentences have been tokenized before transforming them into vectors.

Methods

There are multiple ways to implement intent classification and in this paper, we will be attempting to discuss about the working of the BERT model along with its merits over the state of the art technology that is mentioned in ‘Supervised and Semi-supervised Text Categorization using LSTM for Region Embeddings’ by Rie Johnson and Tong Zhang [6].

We wanted to have a baseline model before we begin with the implementation of BERT methodology in order to better compare the results and outcomes that we obtain.

The four methods that we have explored are:

1. Baseline Model – Simple Sequential Neural Network.
2. BERT Model with Keras Embeddings.
3. BERT model using BERT’s default embedding technique.
4. BERT model using regional embeddings from ELMo.

These methodologies and experiments are detailed further in the next section under experimental work.

Experimental Work

**Approach-1:**

In Approach-1, as a baseline, we have conducted an experiment to generate a text/intent classifier using simple sequential model in Keras. Sequential model is a linear stack of layers.

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

Our baseline sequential model is comprised of an Embedding layer, flatten layer and a dense layer with sigmoid as the activation function which is then compiled with ‘adam’ optimizer by using ‘categorical\_crossentropy’ as the loss function. Each layer takes one tensor as input and outputs one tensor. ‘adam’ optimizer leverages the power to find the learning rate for each parameter. The loss function ‘categorical\_crossentropy’ works well for a multiclass classification.

The first hidden layer i.e., embedding layer converts each sentence into a fixed length vector of defined size. The output from this layer is passed to flatten layer. Flatten layer combines the vectors for a sentence into a single column which is passed to a dense layer. Dense layer applied sigmoid activation function returns the output.

Table

Description automatically generated

*Figure 4.1 – Baseline Model Summary*

**Approach-2:**

In Approach-2 we tried using BERT model with different Keras word embedding technique. This is a different kind of attempt to see how text classification gets done when embeddings from different model are fed to the pre-trained BERT model.

As BERT takes in a specifically formatted inputs, below objects have been created along with the text from the news articles which has been tokenized [2].

* Input ID to correlate the input token with its index number in BERT tokenizer vocabulary.

|  |
| --- |
| 200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249 |

* A segment mask to identify the presence of number of sentences in the article.
* Attention mask to include 1’s for all input tokens and 0 for padding tokens.
* Target variable ‘Class’ values have been encoded to 0, 1, 2 and 3 which corresponds to World, Sports, Business and Sci/Tech.

Target values are assigned from 0 instead of starting from 1 due to the CUDA runtime errors which made the labels incompatible to be accommodated into the model.

Padding and truncating of the input sentences is mandatory owing to the fact that the BERT needs the input arrays to be of the same size. Padding is done towards the end of the sentence and not at the beginning.

Keras sequential is composed of single embedding layer and compiled using ‘adam’ optimizer. This model is used to generate the embeddings for sentences. The embeddings are flattened to generate a single vector for each sentence.

Table

Description automatically generated

*Figure 4.2 – Summary of Sequential Model used for Embedding*

The embeddings generated above are fed to BERT pre-trained model for classification. The model outputs a vector of 728 numbers (base version) which are used for the intent classification.

In this experiment we split above embeddings into train and validation and are further used to generate the tensors and iterators to fine-tune the BERT model. We use a normal BERT model “*BertForSequenceClassification*” for our experiment. Below is the summary of the encoder and same summary will be repeated 12 times.

Text, letter

Description automatically generated

*Figure 4.3 – BertForSequenceClassification Encoder Summary*

We will discuss more about the parameter tuning in approach-3, as same parameters are used. After performing training, the model’s performance has been evaluated on the validation data.

**Approach-3:**

Approach-3 is implementation of BERT model using default BERT tokenizer technique. Before tokenizing the sentences, special tokens [CLS] and [SEP] are added to each sentence to mark the beginning and ending.

The sentences are tokenized and are mapped to the index numbers from the unique set of vocab present for our input data using “BertTokenizer”. We have applied padding technique to have all the sentences in same length. To handle the relationship between multiple sentences, we used the technique of attention mask and segment mask as did in Approach-2.

After we tokenize the sentences and convert to ids, we split the data into train and validation sets and create tensors and iterators for fine tuning the BERT model.

We used same “*BertForSequenceClassification*” used in approach-2. Below is the list of parameters defined for hyper tuning.

Graphical user interface, text, application, email

Description automatically generated

|  |
| --- |
| 300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349 |

*Figure 4.4 – BERT fine-tuning parameters*

We get the weights for various layers and put them in a single list. These weight parameters are separated based on bias, gamma and beta. Using these parameters, we set the optimizer with learning rate of 2e-5. The model is trained with torch dataloader and training loss is captured.

We validated the trained model using validation dataloader and model performance is measured.

**Approach-4:**

In Approach 4 we used Elmo (Embedding from Language Models) with the BERT model to classify the text/Article. Language is complex. Context can completely change the meaning of the individual words in a sentence. Elmo. Developed in 2018 by AllenNLP, it goes beyond traditional embedding techniques. It uses a deep, bi-directional LSTM model to create word representations. Rather than a dictionary of words and their corresponding vectors, Elmo analyses words within the context that they are used. It is also character-based, allowing the model to form representations of out-of-vocabulary words.

**ELMo-Architecture**

Elmo word vectors are computed on top of a two-layer bidirectional language model (biLM). biLM model has two layers stacked together. Each layer has 2 passes — forward pass and backward pass. Raw word vectors act as inputs to the first layer of biLM. Forward pass has information about the word and the context before that word. Backward pass has information about the word and context after it, in a result it forms an Intermediate word vector which is input to the second layer of BilM. The final representation (Elmo) is the weighted sum of the raw word vectors and the 2 intermediate word vectors. Unlike Word2vec and glove word embeddings, the Elmo vector of a token or word is a function of the entire sentence containing the word. Therefore, the same word can have different word vectors under different contexts. Elmo word vectors successfully address polysemous words.

**Implementation of Elmo for Text Classification**

|  |
| --- |
| 350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397  398  399 |

We have used tensor Hub to access Elmo in our implementation. Every word in the input sentence has an Elmo vector of size 1024. To arrive at the vector representation of an entire tweet, we have taken the mean of the Elmo vectors of constituent terms or tokens of the tweet. We ran out of computational resources (memory) to extract embeddings for the articles in one go. As a workaround, we split the train into batches of 10 samples each. Then, passed these batches sequentially to the function elmo\_vectors( ) and concatenate them back to a single array. We have used these Elmo vectors of the dataset to build a classification model Which is BERT. However, this experiment stood as incomplete owing to the CUDA device-side assert triggered error. Made attempts in resolving the issue but none were successful and hence were not able to compare the results of this model with other experiments.

Results

We used metric “Accuracy” to understand the performance of the model and compare against others.

|  |  |  |
| --- | --- | --- |
| **Approach** | **Method** | **Accuracy** |
| 1 | Baseline – Keras Sequential Model | 0.24 |
| 2 | BERT with Keras embedding | 0.25 |
| 3 | BERT with default embedding | 0.88 |

*Table 5.1 – Comparison of Models Accuracy*

From the table above, both baseline model and BERT model with Keras embedding performed poor to classify the sentences into their correct class. While the approach-3 BERT model gave better result than rest approaches.

Keras sequential embedding requires the input data to be integer encoded. As we performed one-hot encoding of the sentences which ignores the context, the ability of the model to classify the sentence is poor. Approach-2 is a kind of experiment where Keras embeddings are passed to BERT pre-trained model. As the embedding of the sentences are created using one-hot technique, the model classification is poor.

|  |
| --- |
| 400  401  402  403  404  405  406  407  408  409  410  411  412  413  414  415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437  438  439  440  441  442  443  444  445  446  447  448  449 |

In approach-3 BERT model generates embeddings based on the context, which means same word can have different vectors based on the context in which the word is used. As BERT with inhouse embedding technique is a context-dependent it performed well better than rest approaches in classifying the sentences into correct classes.

Future Scope

* Ensemble learning to effectively use the best of multiple embeddings.
* Due to available GPU resources in Colab, data size is limited. Models can be larger datasets with more GPU resources.

References

[1]-Kana, M. (2019, October 12). Natural Language Understanding with Sequence to Sequence Models. Medium. https://towardsdatascience.com/natural-language-understanding-with-sequence-to-sequence-models-e87d41ad258b

[2]-Kana, M. (2020, June 15). BERT for dummies — Step by Step Tutorial - Towards Data Science. Medium. https://towardsdatascience.com/bert-for-dummies-step-by-step-tutorial-fb90890ffe03

[3]-Krish Naik. (2020, May 8). Word Embedding - Natural Language Processing| Deep Learning. YouTube. https://www.youtube.com/watch?v=pO\_6Jk0QtKw

[4]-AG’s corpus of news articles. (n.d.). AG\_corpus\_of\_news\_articles. Retrieved June 11, 2021,from http://groups.di.unipi.it/%7Egulli/AG\_corpus\_of\_news\_articles.html

[5]-mhjabreel/CharCnn\_Keras. (n.d.). GitHub. Retrieved June 11, 2021, from https://github.com/mhjabreel/CharCnn\_Keras/tree/master/data/ag\_news\_csv

[6]-Rie Johnson, Tong Zhang. (2016, May 26). Supervised and Semi-supervised Text Categorization using LSTM for Region Embeddings

<https://arxiv.org/pdf/1602.02373.pdf>

[7]-A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text - https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

|  |
| --- |
| 450  451  452  453  454  455  456  457  458  459  460  461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483  484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499 |