COMP-1818 Artificial Intelligence Application Report

Abidon Jude Fernandes
af3761m@gre.ac.uk
001013672

2020.12.11

Abstract

This report is for the submission of my COMP1818 artificial intelligence applications module. In this report I will be discussing my research side of the sentiment analysis part and my team contribution in this coursework group project. In this paper I will explain the idea behind our project, go through all the steps for my algorithm model and why I used these specific models. Keywords (RNN, LSTM, GRU, BiLSTM, BiGRU, Overfitting and Natural Language Processing)

1 Introduction

1.1 Research

The authors mentions that on the internet there are large amount of unstructured data full of opinions and reviews. These opinions and reviews are not fact checked against a functioning algorithm so there are multiple instances of false information being spread to multiple people. These information can change a person emotion depending on the text that is being portrayed to them. This can affect the company and the person as a whole when it comes to differentiating between true information and false information. Natural Language Processing is used to extract useful information from an unstructured data by converting it into a structured form. They performed a sentiment analysis into customer reviews which did not have any rating system to find out if the deep learning algorithm is capable at determining a review to be positive or negative. Neural network often fail to capture contextual meaning of words and fails to save long sequences of words and it results in a reduce performance. The solution to this problem is a hybrid model such as RNN-LSTM-BiLSTM-CNN. in a pre-trained word2vec and Glove embedding (100d). The author found problems in the recurrent neural network. The problem is it cannot save previous information if the gap is small between the current and previous information which is a vanishing gradient problem. To solve this issue a BiLSTM model was used because of its ability to read sequences backwards and forwards whereas the normal LSTM is not able to consider any future sequences of a hidden unit while calculating the output. The result of this experiment proved to be successful because the Bidirectional model and a pre-trained word embedding achieved high accuracy along with the author proposed model (Aslam et al. 2020).

1.2 Proposal

Our group topic was sentiment analysis with tweets using RNN. We selected to use the airline sentiment data and went on our own to do research. My goal after reading all of my scientific papers on sentiment analysis is to create a BiLSTM and a BiGRU recurrent neural network model using a pre-trained GloVe word embedding using a Twitter data set full of uncleaned tweets. The reason why I decided to take this route is because the idea of using a natural language processing deep learning algorithm such as recurrent neural network with an uncleaned airline Twitter data set one of my colleagues proposed was a great idea when it came to testing out the bidirectional LSTM and GRU in a text classification problem to differentiate positive or negative

tweets. My idea differentiate from my colleagues because one of them is researching the pre-trained word embedding layers which are Word2Vec and GloVe and will be creating a matrix layer while the other will go more in-depth on the theoretical side of the recurrent neural network and produce a simple LSTM and GRU model for comparison. My proposed algorithm will go as follows. The Twitter data set full of negative, neutral and positive tweets will be taken from the Kaggle website domain `https://www.kaggle.com/crowdflower/twitter-airline-sentiment`. I will import all the necessary libraries for my entire program to function properly. The system will work as follows, the data set will get read through a pandas data frame then the text and the sentiment results will be taken taken out and placed into a new data set. The neutral sentiment result will be removed because i want the model to only determine what is positive or negative. The text pre-processing stage will clean the entire tweet and later it will be tokenised, placed in a sequence then padded so it is able to be fed into a pre-trained GloVe embedding layer. After it is trained with the vectors it will be transferred to a Bidirectional training model for training. The model will be tested in a testing class which has an array of multiple negative and positive texts. The same data set will be loaded in and the trained model will be used to see if it can predict accurately on an unseen data set. The overall result of my experiment was successful after 10 epochs both my Bi-LSTM and Bi-GRU model was able to detect positive negative tweets. Bi-GRU was more fast than BiLSTM since it was able to train quickly because it has less trainable parameters. The pre-trained word embedding and the amount of model layers dominated the result output. The knowledge I have obtained during my implementation stage was a lot. I am now able to load a data set, clean the text, go through the tokenisation process and train the model at ease. My methods came to because of my colleagues suggesting I use a pre-trained word embedding to increase my accuracy and another suggested I use multiple layers to get a better accuracy in my model which I did implement for a better result.

2 Methods

2.1 Models and algorithms

The models we created in our coursework were Word2Vec and GloVe word embedding, LSTM and GRU training model and Bidirectional LSTM and GRU. I created the Bidirectional LSTM and GRU training models and used a pre-trained word embedding layer to further enhance the accuracy of my training models and at the end I tested the model using a string array of texts and the same airline sentiment Twitter data set. LSTM was proposed to solve the vanishing gradient issue where the vanilla RNN and forward feedback neural network failed to remembers certain words because of the information gap. It makes this up by having a long dependencies capable at remembering information fro long periods. It has two mathematical gates such as tanh which regulates the information at a value of -1 or 1 and sigmoid which outputs 0 or 1 value. The three gate it has are input, forget where it decides which information it shall keep or dismay and the output gate. (Hochreiter and Schmidhuber 1997). Gated Recurrent Unit is an improve model of the LSTM network. It uses two gates which is the update gate that performs the job of filtering the information and updating it and the rest gate which rests the cell state. It is much more quick than the LSTM and has less trainable parameters but it suffers in accuracy since it has only two gates fused together (Chung et al. 2014). In the end both these two basic RNN LSTM and GRU suffer from remembering the output of a sequence so a bidirectional RNN is used where it can go forward and backwards when reading a text.

2.2 Experimental settings

The experimental setting went as follows: Import all the necessary libraries I need and those libraries are pandas for loading in the Twitter airline sentiment tweets, numpy to fro general array handling, tensorflow and keras for calling the BiLSTM and BiGRU model features and for the tokenisation process, NLTK for the text pre-processing stage, matplotlib.pyplot for charts and graph creation, re for opening files string for text usage and sklearn for splitting the data set and

general accuracy testing. The second stage was to load the Twitter airline sentiment data set in a Panda data frame and noticed they were a lot of unclean tweets and unnecessary sentiment results I do not need. To fix this issue I created another Panda data frame which contains only the Twitter tweets and positive and negative airline sentiment text. The positive text got assigned with an integer value of 1 and the negative text got assigned with the integer value of 0 because the model only accepts integer values. The third stage was the text pre-processing stage where we clean the text of all the unnecessary feature it has and at the end the text should be lower cased and clean. To clean the text of any unaccepted emojis, punctuation's and stop words. I create a class which removes all the emoji pattern using Unicode. It removes emoticons, symbols and pictographs, transports and map symbols and FLAGS (iOS). Afterwards I created a class for removing punctuation's and stop word class to remove stop words using the NLTK library, split the text and lower case all of it. Once the pre-processing stage is finished i used the sklearn library to split the data to 80 percent for training and 20 percent for testing. The fourth stage was the tokenisation stage in which i used the Keras and Tensorflow library to handle this. The hyper parameters are vocabulary size at 10000 words, out of vocabulary token at "OOV", max text length at 280 since that is the maximum length of each tweet in the year 2020, truncating type at post and padding type at post. I tokenised the training data and printed out the result. This showed me the word index is a dictionary showing you the key being the word and the value being the token for each word it is assigned to. Any out of vocabulary text were fitted with the "OOV" text and were included into the index. The sequence stage creates sequences of tokens representing each sentence. The padding stage pads the sequence into an array. The padded sequence have multiple zeros in the front because some of the indexes are large and it applied zero to all the rest to have an equal size. To solve this I made it so it obeys the max length of 280, set padding to post where it will make the zero appear last and use the truncating post function to chop off any work that goes beyond the max length. Now I took a pre-trained GloVe word embedding model which one of my team mates have done and applied it to my model and this improved the final model accuracy by a lot. This word embedding layer encodes the words into vectors because it uses occurrence count as to transform words into meaning because of its linear substructures (Pennington, Socher, and Manning 2014). The recurrent neural network models I used were BiLSTM and BiGRU. Tensorflow and Keras library was used to import these models and its associated layers. Both the model had similar structure. The BiLSTM and BiGRU model had an embedding layer which connects to the GloVe model and training was set to false so it does not train while the model is running, in the embedding layer it has the embedding dimension set to 100, number of words, max length and the embedding matrix. The model itself has a 2 dropout layer. A spatial dropout 1D set at 0.2 after the embedding layer and a dropout layer at 0.25 after the second model layer. I used two layer fro max efficiency for my training data. The first contains a BiLSTM and BiGRU mode with 64 nodes and set to True for the return sequences in the hidden layer and the second one has a 32 node. I added two dense layer, one was 32 set to relu and the last one was a sigmoid function set to 1. I use binary crossentropy for loss, adam for optimiser and accuracy for metrics (Martın Abadi et al. 2015). The embedding layer is used to set up the model in which the text is being processed into before going into he mode itself for classification. The model has nodes and layers to process the text and outputs its result in a vector size. The dropout is used to prevent overfitting, it does this by randomly setting the input units to zero with a frequency rate at each step. The dense layer is used to feed the outputs from the previous layers to all it neurons. The relu mathematical function gives an output one if the unit is positive and zero if the unit is negative. The sigmoid function is a predicting probability function calculator. The was tested with 10 and 100 epochs with the 1 verbose and the validation data set to the split test data. mathplotlib.pyplot was used to display the graphs and finally, a testing class with muitlpe string arrays was used to see if it can predict the text emotions and the same sentiment data set.

2.3 Evaluation criteria

We evaluated the model by looking at the validation accuracy and validation loss to see if it was underfitting or overfitting against each other. Later we came will come to the conclusion on which model was the best in terms of accuracy. We unfortunately had no time to create a precision, F1 score and recall accuracy test for our model result.

3 Results

The result for the BiLSTM at 10 epochs has the loss of 0.18 and accuracy at 0.92. The validation loss at 0.27 and the validation accuracy at 0.91. The BiGRU at 10 epochs has the loss of 0.282 and accuracy at 0.93. The validation loss was at 0.27 and the validation accuracy at 0.9135. At 100 epochs however the BiLSTm started to overfit at 18 epochs whereas the BiGRU started to overfit at 13 epochs.

4 Discussion

The 10 epochs results showed up that BiLSTM performed slow but better in detecting emotions when it came to accuracy but the BiGRU was quicker and had a minor accuracy loss in difference when compared to BiLSTM. In the end I and my team went with BiGRU since it was quicker. The 100 epochs test on both of these model showed us that overfitting occurs in a complex model built when paired with a small data set since the model does not learn but memorises the accuracy and the loss increases with it. Our data set was skewed because it was not balanced and the model did not have an equal amount of information on both the positive and negative side to train because most of the information was favouring the negative side.

5 Conclusion

In conclusion, I learned a lot in this module and we as a group did well in creating a natural language processing model. In the end we chose the BiGRU model to be our success model when detecting emotion in tweets because it is fast and useful for a small data set.

In future works, I will learn and create an accuracy score to determine which model performed better instead of relying on the testing string array model for evaluating models. Due to page restriction I apologise I cannot show you all my fully experimented results but in the future for other projects I will include pictures of my training and testing results.

Contribution

I would like to thank these two colleagues of my for helping me and each other when it came to creating this coursework.

Ashley James Walker taken the lead role in suggesting the idea of sentiment analysis and researching heavily into the theoretical side of this algorithm and creating simple a LSTM and GRU.

Sean Daniel Daley taken the role of researching and creating the pre-trained word embedding layer.

References

Aslam, A. et al. (2020). "A Novel Framework For Sentiment Analysis Using Deep Learning". In: *2020 22nd International Conference on Advanced Communication Technology (ICACT)*, pp. 525–529. DOI: 10.23919/ICACT48636.2020.9061247.

Chung, Junyoung et al. (Dec. 2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In:

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Martın Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.