Deep Learning (COSC 2779) – Assignment 2 – 2021

Keavatey Srun (S3767615)

1 Problem Definition and Literature Review

Considering there is a source of Twitter and a set of replies discussing the rumours. The task is to classify the replies into either supporting, denying, questioning, or commenting on the underlying rumours initiated by the source tweet. Thus, the problem is called stance classification on the replied texts, or simply put it is a text classification problem on Twitter data with multi-outputs.

On the submission to SemEval 2017 RumourEval, the proposed model was an LSTM based sequential model for this specific task with an accuracy of 0.782 and Macro F of 0.561 on development data and an accuracy of 0.784 and Macro F of 0.434 on testing data (Kochkina, Liakata and Augenstein, 2017). The paper used the branch LSTM model with the word2vec model pre-trained on the Google News dataset. There is a gap between macro F1-score on development data and testing data due to different class balances in these sets, which means the model did not generalize well. The model could only capture commenting, the majority class, but failed to predict the other three classes, the minority classes.

At RumourEval 2019, different machine learning algorithms were used by different teams. A gradient boosting classifier stood out among the traditional ML approaches. However, the deep learning approach using LSTM based took first place instance classification. By considering the skewness towards the comment class, macro-averaged F1 was employed to evaluate the performance of the task. The best three teams used word2vec model pre-trained on the GoogleNews dataset (300d) and reached Macro F1 score between 0.5776 and 0.6187 (Gorrell et al., 2019). The fourth-place achieved a macro F1 score of 0.4895, and the fifth-place achieved macro F1 of 0.4792.

Another paper at SemEval-2019 used the Bi-LSTM network to predict the same problem and achieved 0.435 F1-Macro in the stance classification task (Hamidian and Diab, 2019).

2 Evaluation Framework

Since this is a multi-class classification problem, and the outputs was integer encoded before fitting into tensor, categorical cross-entropy was considered the most appropriate loss function for the model

The dataset is highly imbalanced, so accuracy will not make much sense to determine if the model captures all classes well enough. F1-score is a good metric for an imbalanced class since it takes both precision and recalls into account. Plus, the F1-score macro average weights the metric towards the smallest class. According to the literature review, macro-averaged F1-score was also used as the performance metrics in the same classification task. The macro F1-score from RumourEval 2019 on the stance classification task among the top 5 teams was between **0.4792 and 0.6187**. Therefore, the target F1-score of this task was expected to be within this range.

3 Approach & Justifications

- Data pre-processing

The problem of this assignment is quite different from the normal sentiment analysis because the task is to classify the stance of replies attached to a specific source of Twitter text. To achieve that goal, we needed to pre-process the dataset to have each set of replies attached to the same source of Twitter (Appendix A).

After pre-processing, the final dataset consists of 6,253 rows. The target has four classes. As seen in Figure 1, it is highly imbalanced among the four classes. The majority goes to "comment", which is more than 50%. Other classes are the minority. There is a need to deal with this highly imbalanced dataset. Otherwise, the model would not capture all classes well. Thus, up-sampling with text translation was used, which could be referred to as text augmentation. By up-sampling, more data points

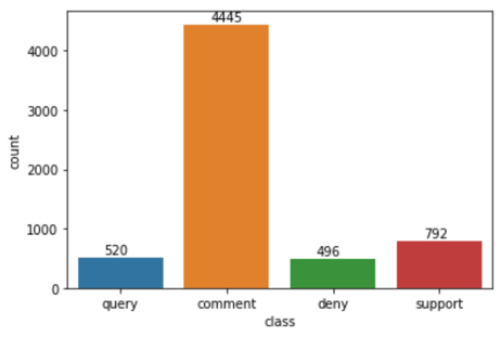


Figure 1: Bar plot of reply class

corresponding to the minority classes were injected into the dataset, so it is balanced between each class. Also, translating replied texts into random foreign languages and translating them back into English helped increase the number of words by having new words in the dataset.[1]

- Text cleaning and tokenizing

The target variables were encoded with integer encoding. Then, text cleaning was performed using the following methods on both replies and source:

- removing URL
- converting emojis to text
- removing mentions (@)
- converting text to lower case
- removing rt as it means retweet
- removing remove all punctuations in source while removing punctuations, except "?", "!" and "'" in replies considering they might be useful in classifying the class
- removing stop words as they do not contribute much to text classification

Then tokenizing was performed on the cleaned text. Since the lengths of the text were not the same, post-padding was used to make all sentences in the same length.

Finally, the data was split into train, validation and test set based on 80:10:10 ratio.

- Embedding and model building

Since the dataset is sequential data and the goal is to classify the text into four outputs, branch LSTM was employed. This architecture was selected based on the input from the literature review as well.

Transfer learning using a pre-trained Glove on Twitter data with 200 dimensions or pre-trained Word2Vec on GoogleNews dataset with 300 dimensions was implemented on both source and reply before merging them together and fitting into the LSTM model. Bidirectional LSTM was used since it could use both past and future information to classify the data.

4 Experiments & Tuning

by following the same steps to fit and fine-tune the hyperparameters, we did two experiments to see which yielded the best result.

- Model 1

After the initial experiment, the first model was fitted with just two bidirectional LSTM layers with 100 units and two dense layers with 64 units and 32 units, relu as activation function and l2 regularization (lambda = 0.0001). Because the output is multi-classes, the last layer is a dense layer with 4 hidden units and softmax as activation. This layer stayed the same throughout. The result showed it was very overfitting. Therefore, the dropout layer and learning rate were added in the next model.

- Model 2

Model 2 consists of the two bi-LSTM layers, which is the same as model 1. Next is a dense layer and a dropout layer with a 0.3 dropout rate. Then another dense layer and dropout layer with the same dropout rate. The number of hidden units in both dense layers before the last one was increased to 100. The change of the hidden units aimed to increase the capacity of the model to increase the F1-score. The lambda of l2 regularize was changed to 0.1. Optimizer is changed from Adam to RMSprop to see any improvement when switching to a different optimizer. A learning rate of 0.0003 was also added to prevent overfitting.

- Model 3

It is the same as model 2, with an increase of dropout rate to 0.5 to see if it helped reduce overfitting.

- Hyperparameter Tuning

After noticing the improvement of the third model, we decided to use it as the base model for parameter tuning. Three hyperparameters, along with the reasons below, were used to tune for the hyperparameters of the model. A random search was used to pick the best values among the selected values based on each parameter's observations.

[1] The code to implement this is extracted from this GitHub medium-article-code/data_augmentation_using_language_translation.ipynb at master · kothiyayogesh/medium-article-code · GitHub. However, there was an error and was fixed using the solution from here https://stackoverflow.com/questions/69194625/textblob-ocr-throws-404-error-when-trying-to-translate-to-another-language. That was why this part of task was done in a separate notebook through PyCharm.

- Learning rate = [1e-3, 3e-3, 1e-4, 1e-5]: learning rate is critical in model performance. There is a need for a proper learning rate to have an optimal model. Too high or too low a learning rate would result in a model with low effective capacity.
- Lambda value of l2 regularization = [0.1, 0.01, 0.001, 0.0001, 0.0003]: regularization is used to prevent overfitting.
- Dropout rate = [0.25, 0.3, 0.5, 0.75]: dropout rate is also used to prevent the overfitting of a model.

4.1 Experiment 1: Training data with text augmentation and pre-trained Glove

For the first experiment, we fitted the model on the original dataset using pre-trained Glove. That means the imbalance class still existed in the dataset. As shown in Figure 2 (Appendix B), the first model was very overfitting. After adding dropout and changing other values, the performance of the second model improved. The gap between the training set and the validation set was reduced. The performance was further enhanced once the third model was fitted.

4.2 Experiment 2: Training data with text augmentation and pre-trained word2vec

For the second experiment, we fitted the model on the dataset after text augmentation using pre-trained word2vec on GoogleNews dataset, considering it was used in the literature review. In Figure 3 (Appendix B), the three models behaved the same way in the first experiment. However, the overall loss value was slightly higher, and the F1-score was slightly lower than that in the first experiment. That indicates the pre-trained word2vec did not make any significant progress in the model performance.

4.3 Comparing both experiments

Random search results showed that the same values of each parameter were selected in both experiments: learning rate of $1.e^{-5}$, lambda value of 0.01 and dropout rate of 0.3. Based on Figures 4 and 5 (Appendix C), the same value of the parameters model in experiment 1 yielded a higher result. The F1-score was higher (over 0.5) than in experiment 2 (lower than 0.5). That means the pre-trained model on Glove as an embedding provided a better result; thus, it was chosen over the pre-trained word2vec model.

5 Ultimate Judgment, Analysis & Limitations

After deciding on the pre-trained embedding and value of hyperparameters, we used the model and values to determine the final best model for the task. We found the optimal model after a few iterations on Adam and RMSprop as an optimizer and number of epochs to train. The final model was complied with Adam of optimizer and trained with 20 epochs. It yielded an F1-score of 0.5271 on the training set and 0.5231 on the validation set, which means the model is well-fitted. This F1-score is within [0.4792, 0.6187]; hence, the goal was accomplished.

When used to evaluate the final selected model on the unseen test data, the F1-score is not much different, which is 0.53, slightly higher than that on the validation set. It can be assumed that the model can generalize well on the unseen data. From Figure 7 (Appendix C), the classification report shows that the "query" class has the highest precision and F1-score, followed by the "deny" class. The other two classes have a similar score of precision and F1-score.

There are some limitations of the model. First, although the F1-score reached the target range, it did not surpass the top 3 highest score. Second, only three parameters were fine-tuned. Other parameters like hidden units in the dense layers were not properly fine-tuned. Last, we did not get to experiment with the pre-trained BERT embedding. Future studies should address the last two limitations to see if the model's performance could be improved and surpass the scores of the top 3 teams that participated in RumourEval 2019.

6 References

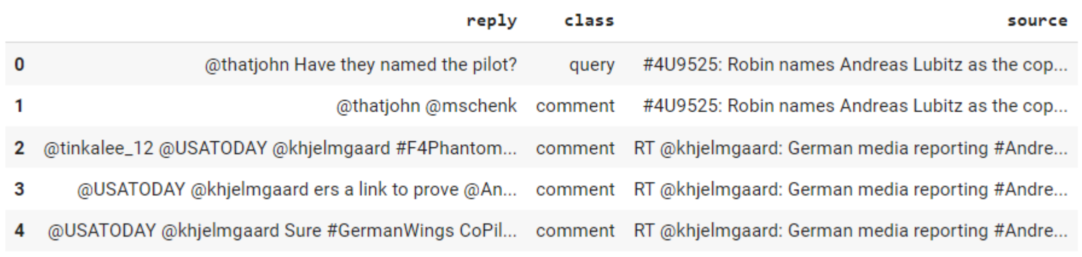
Gorrell, G., Kochkina, E., Liakata, M., Aker, A., Zubiaga, A., Bontcheva, K. and Derczynski, L., 2019. SemEval-2019 Task 7: RumourEval, Determining Rumour Veracity and Support for Rumours. *Proceedings of the 13th International Workshop on Semantic Evaluation,* [online] Available at: <https://aclanthology.org/S19-2147/> [Accessed 10 October 2021].

Hamidian, S. and Diab, M., 2019. GWU NLP at SemEval-2019 Task 7: Hybrid Pipeline for Rumour Veracity and Stance Classification on Social Media. *Proceedings of the 13th International Workshop on Semantic Evaluation*, [online] Available at: <https://aclanthology.org/S19-2195/> [Accessed 11 October 2021].

Kochkina, E., Liakata, M. and Augenstein, I., 2017. Turing at SemEval-2017 Task 8: Sequential Approach to Rumour Stance Classification with Branch-LSTM. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, [online] Available at: <https://aclanthology.org/S17-2083/> [Accessed 11 October 2021].

7 Appendix

Appendix A: dataset after pre-processing to each set of replies attached to the underlying source tweet



	reply	class	source
0	@thatjohn Have they named the pilot?	query	#4U9525: Robin names Andreas Lubitz as the cop...
1	@thatjohn @mschenk	comment	#4U9525: Robin names Andreas Lubitz as the cop...
2	@tinkalee_12 @USATODAY @khjelmgaard #F4Phantom...	comment	RT @khjelmgaard: German media reporting #Andre...
3	@USATODAY @khjelmgaard ers a link to prove @An...	comment	RT @khjelmgaard: German media reporting #Andre...
4	@USATODAY @khjelmgaard Sure #GermanWings CoPil...	comment	RT @khjelmgaard: German media reporting #Andre...

Appendix B: Experiment

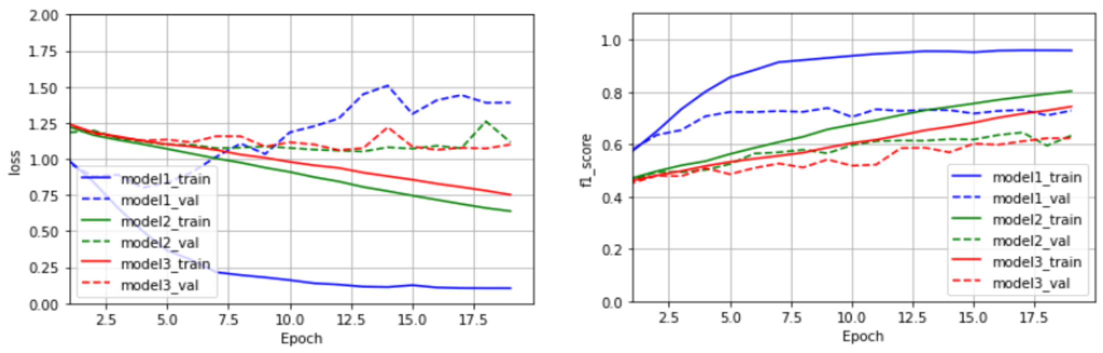


Figure 2: Plot of the first three models on text-augmented data with pre-trained Glove

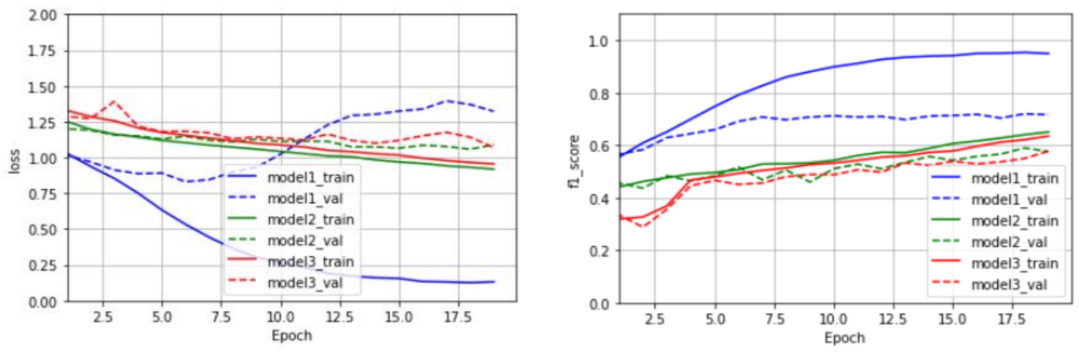
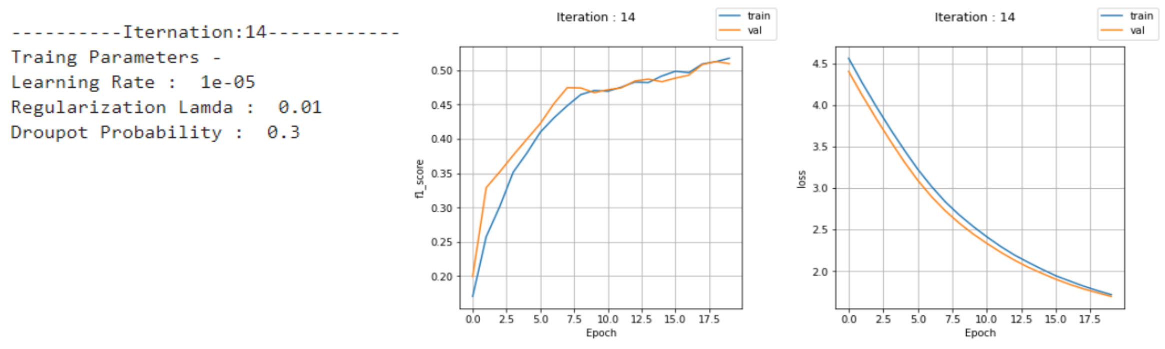


Figure 3: Plot of the first three models on text-augmented data with pre-trained Word2Vec

Appendix C: Result

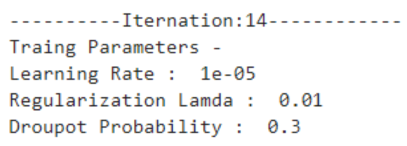
- The best model with parameters selected from the random search on experiment 1



```
----------Iternation:14------------
Traing Parameters -
Learning Rate :  1e-05
Regularization Lamda :  0.01
Droupot Probability :  0.3
```

Figure 4: Plot of results of the random search on experiment 1

- The best model with parameters selected from the random search on experiment 2



```
----------Iternation:14-----------
Traing Parameters -
Learning Rate :  1e-05
Regularization Lamda :  0.01
Droupot Probability :  0.3
```

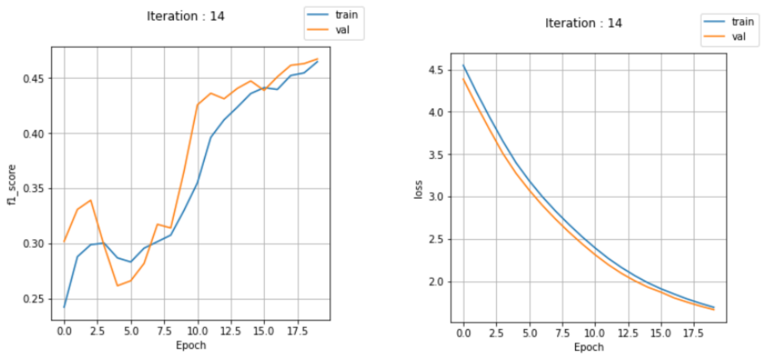


Figure 5: Plot of results of the random search on experiment 2

- The best model on data with text augmentation using pre-trained Glove

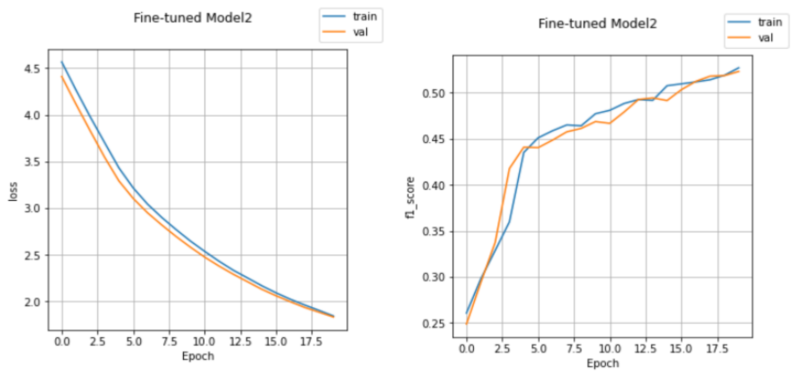
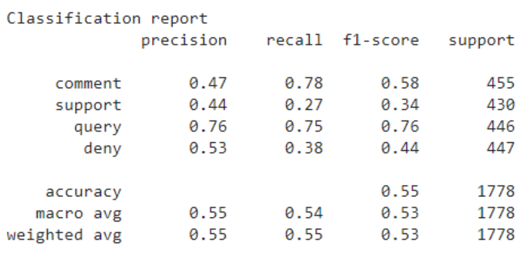


Figure 6: Plot of the best and final model

- Evaluation of the final model on the test set



```
Classification report
              precision    recall  f1-score   support

     comment       0.47      0.78      0.58       455
     support       0.44      0.27      0.34       430
       query       0.76      0.75      0.76       446
        deny       0.53      0.38      0.44       447

    accuracy                           0.55      1778
   macro avg       0.55      0.54      0.53      1778
weighted avg       0.55      0.55      0.53      1778
```

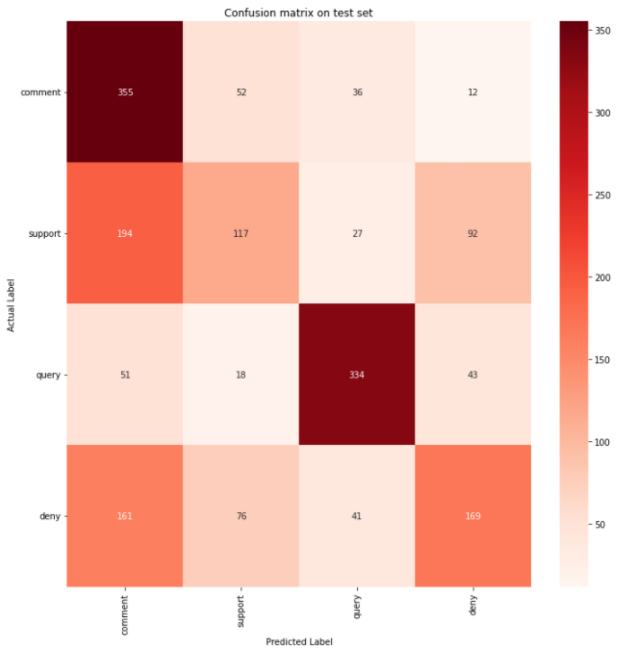


Figure 7: Classification report and confusion matrix of the final model on the test set