Deep Learning Project - Rumour Veracity and Support

Name: Nikhil H

1 Problem Definition and Review

To classify the response tweets into the corresponding category of support, deny, query or comment based on the source tweet.

Data set Analysis:

The rumour dataset has 6 attributes and there are 6253 response tweets and 381 source tweets. The response tweets are grouped into 4 categories SDQC (Support, Deny, Query, Comment).

Data Exploration:

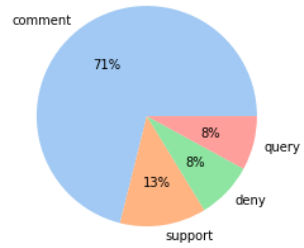

Figure 1

Exploration of feature Class: From the pie chart in Figure 1 we can clearly see that the number of records for the class comment is more than the other classes. So that dataset is imbalanced. The value = -1 which corresponds to source msg is excluded.

Exploration of the feature Text: The tweet text consists of special characters and alphanumeric characters and URL. So, all these will be removed from the data.

Exploration of the feature Source_Msg_ID: Each of the response tweets correlates to the source tweet by the Source_Msg_ID.

2 Evaluation Framework

Since I have oversampled the data to solve the data imbalance, I am considering the accuracy as the performance metric. The below results are evaluated on the test set which corresponds to 20% of the twitter dataset.

Model Name	Parameter	Value
Model A (comment classification)	Loss = Binary Cross entropy	0.67
	Performance Metric = Accuracy	64 %
Model B (SDQ Classification)	Loss = Categorical cross entropy	0.98
	Performance Metric = Accuracy	71%

3 Approach & Justifications

1) Explore the dataset to check for class imbalance.
2) Pre-process the tweet text by converting all the text to lower case, remove all the punctuations, strip the white spaces, and remove the alphanumeric characters and the URL's.
3) Concatenate the source and a response tweet and add a separator ('|') between the tweets. The separator '|' will be encoded as 1064 by the Bert encoder. So, the dataset will have only one feature Text (Source tweet | Response tweet) and the target attribute Class. The other features will be dropped.
4) Since the dataset is too imbalanced, I will approach this problem in two stages. The first stage consists of training the model to classify response tweets into comment and non-comment (Model A). The second stage consists of classifying the response tweets into SDQ (Model B).
5) Make a copy of the pre-processed data and create two new data frames which will be used by Model-A and Model-B.
6) To train Model-A I will encode the records with value comment = 1 and other records as non-comment = 0. Still the dataset is imbalanced as it has 4445 records for comment class and 1808 records for the non-comment class.
7) For training Model-A I will split the data into train (60%), test and validation. Then, I will oversample the 2600 records for the minority class(non-comment) in the training data. So, the training data contains 2666 records for comment and 2600 records for non-comment. The validation and test data will not be oversampled.
8) I have selected the model Electra-base for training the Model-A and Model-B. The optimizer will be AdamW since it minimizes the prediction loss and does regularization by weight decay.
9) The EarlyStopping is used with the callbacks and the val_loss is monitored, and the patience = 15.
10) A base model is created using Electra with the corresponding pre-processing layer and transfer learning is used to train the Model-A.

11) Add a dense layer to the base model with one node to classify the tweets into comment and non-comment. I will use accuracy as performance metric and binary_crossentropy as the loss. As the data imbalance issue is resolved using oversampling, so I am using accuracy as the performance_metric.
12) Build a complex model with more hidden layers and experiment with the learning rate.
13) Experiment with different activation functions (relu, elu, tanh) and select the best activation function based on the reduce in overfitting and accuracy.
14) Reduce the overfitting in the model by adding a dropout layer. After the model is fine-tuned, use max_epoch = 60 and train the model-A to classify comment and non-comment. Then evaluate the fine-tuned model on the test data.
15) Train the Model-B to classify the tweets into SDQ. From the pre-processed data exclude the comment class records. Then one hot encode the remaining records which correspond to support, deny and query.
16) Split the dataset into train and test and I will continue the same procedure and experiments from step 8 to step 14 for training Model-B. But the difference is that the Model-B will have a dense output layer with 3 nodes for classifying the tweets as Support, Deny, Query. I will use accuracy as the performance metric and categorical_crossentropy as the loss.
17) I have used the reddit dataset for the independent test set prediction. First, I will use the model-A (comment classification) on reddit dataset. This model predicts either 1 for comment or 0 for non-comment. Use the records with prediction = 0 (non-comment) as the test set input for model-B.
18) The Model-B predicts 3 outputs support=0, deny=1, query=2.
19) Since the test data (reddit dataset) is imbalanced I will use F1-score as the performance metric.

Justifications:

Model: I am using the transfer learning approach to train both the models (Model-A and Model-B). I have used the pre-trained model Electra-base because it has much better accuracy than the small Bert models. The Bert models are trained used the masked language modelling (MLM). Where the input tokens are corrupted by replacing it with a mask and then trained to reconstruct the original tokens. So, the model just learns for the masked tokens. In the case of Electra, the inputs are corrupted by replacing some tokens with the samples from the output of a small masked language model. Then the network is pre-trained as a discriminator that predicts for every token whether it is an original or a replacement. So, the network is trained on all the input tokens and hence it has much better accuracy.

Early Stopping and the number of epochs: The number of epochs is set to 60 for the fine-tuned model. For the Model-A since it classifies only comment and non-comment it starts overfitting. So, I have used Early stopping with patience = 15 and monitor = val_loss. So the model waits for 15 more epochs and if there is no decrease in the val_loss then model training is stopped before the max_epochs(60) is completed. During the training of Model-A, the Early stopping was triggered, and it stopped at epochs = 39.

Activation functions:

Output layer for Model-A Sigmoid: The sigmoid is used as an activation function for Model-A (classification of comment and non-comment) as it is a binary classification with a threshold of 0.5.

Output layer for Model-B SoftMax: For the output layer softmax is used as an activation function since we want the probability (max likelihood) of an observation (tweet) to be predicted as one particular Class (Support, Query, Deny). So, this is a multiclass classification problem.

Metrics:

Loss – (Model-A) binary cross entropy: The binary cross entropy is used as the loss function, as the Model-A predicts comment (1) or non-comment (0). So, it is a binary classification model.

Loss - (Model B) categorical cross entropy: The categorical cross entropy is used as the loss function for Model-B as it is a multiclass classification problem (Classification of SDQ) and the target values are one-hot encoded.

Optimizers – Adam(both model-A & model-B): Electra is trained on the same hyperparameters as Bert. But the difference is that Electra was trained using LAMB optimizer instead of adam as it requires less fine tuning.

But LAMB optimizer did not outperform Adam when Adam is fine-tuned. So, I am using Adam as I am fine-tuning optimizer with different learning rates and using the same schedule as the Bert pre-training.

4 Experiments & Tuning for Model-A and Model-B:

Experiments	Tuning	Effect
Creating Simple Model	Use the pre-trained Electra-base model with the corresponding pre-processing layer and add dense layer with 1 output node (Model-A) and 2 output nodes (Model-B).	The baseline model helps in comparing the results.
Complex model	Add 3 hidden layers to each of the model-A and model-B with a greater number of hidden nodes.	Increases the model capacity
Learning rate: 3e-05 vs 5e-05	Use the adam optimizer with lr = 3e-05 and 5e-05.	Helps in selecting the model with high accuracy.
Activation: Relu vs Elu vs Tanh	After experimenting with relu , elu, tanh for both models I have used relu.	There was no significant difference but using the relu both the models have less overfitting.
Dropout	Experiment with different dropout values and I have added 2 dropout layers with each with 0.1 and 0.2 for both the models.	Helps in reducing the model overfitting.

5 Ultimate Judgment, Analysis & Limitations

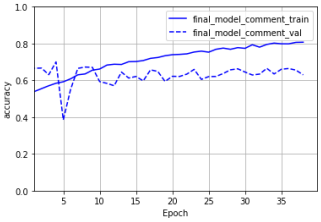

Figure 2

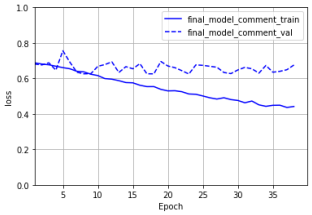

Figure 3

Analysis of Figure 2 and Figure 3:
The Figure 2 and Figure 3 displays the accuracy and loss (binary_crossentropy) for Model A. As we can clearly see that after 25 epochs the model starts overfitting by a small range. But due to early stopping the model training is stopped at epochs = 39.

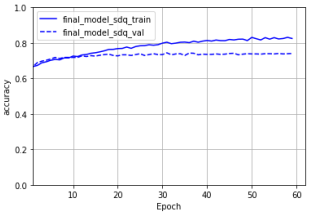

Figure 4

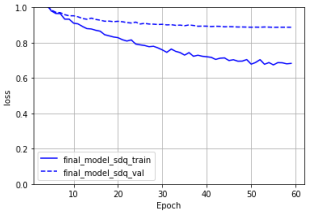

Figure 5

Analysis of Figure 4 and Figure 5:
The Figure 4 and Figure 5 displays the accuracy and loss(categorical_crossentropy) for Model-B. As we can see that after 30 epochs there is a small amount of overfitting.

Testing on Reddit Data:

The reddit data has 2471 records and 6 attributes. The response tweets are categorised into 2250 comment, 94 deny, 64 query, 23 support tweets. So, the dataset is imbalanced.

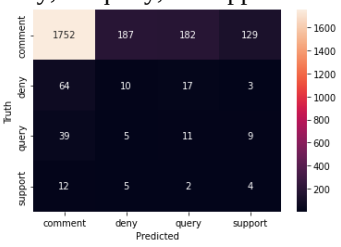

Figure 6

	precision	recall	f1-score	support
1	0.94	0.78	0.85	2250
2	0.05	0.11	0.07	94
3	0.05	0.17	0.08	64
4	0.03	0.17	0.05	23
accuracy			0.73	2431
macro avg	0.27	0.31	0.26	2431
weighted avg	0.87	0.73	0.79	2431

Figure 7

Analysis of Figure 6 and Figure 7:
As the dataset is imbalanced, I will consider the F1-Score as the performance-metric. From the classification report in Figure 7 we can observe that the model is good at classifying comment. But it is not very good at classifying SDQ.

The poor performance on SDQ classification can be related to the smaller number of observation corresponding support, Deny and query. To increase the model performance related to SDQ classification we can gather more tweets which correspond to the Support, Deny and query which can be related to same topics or different topics and retrain the model.

APPENDIX:

Literature Review:

Paper Referenced:	Description of the relevant technique
RumourEval: Determining rumour veracity and support for rumours	This paper addressed the class imbalance issue caused by many records of category comment in the response tweet. So, the classification is divided into two step processes. Where the 1st step involves classifying the tweets into comment and non-comment. In the 2nd step classify the remaining non-comment tweets into SDQ (Support, Deny and Query) [1].
GAN: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples	This paper addressed the Text classification using the GAN (Generative Adversarial Network). The paper discusses about the limitation of Bert like transformers as they produce unstable models when they are trained with few labelled examples. It describes using GAN with Bert to create robust models even with very few labelled examples. [2]
Electra: Pre-training text encoders as discriminators rather than generators	This paper addresses the working of Electra and the parameters used to fine tune the Electra model. The input tokens are corrupted using a small MLM model and then network is trained as discriminator to predict whether a token is original or a replacement. The paper also shows the comparison of Electra-base and other Bert models based on Glue Score. The Electra-base has the highest Glue score with 85.1. The paper also describes about the different optimizers used to fine-tune Electra. [3]

6 References:

[1] Derczynski, L. et al. (2017) "SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours", Proceedings of the 11th International Workshop on Semantic Evaluation

[2] Papers with Code - GAN-BERT: Generative Adversarial Learning for Robust Text Classification with a Bunch of Labeled Examples (2021). Available at: https://paperswithcode.com/paper/gan-bert-generative-adversarial-learning-for

[3] Kevin Clark and Minh-Thang Luong and Quoc V. Le and Christopher D. Manning: ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, ICLR 2020.

[4] Brownlee, J. (2020) Random Oversampling and Undersampling for Imbalanced Classification, Machine Learning Mastery. Available at: https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/