Stance Detection and Rumor Verification

Vaisshnavi Yerrapothu
CSE, University at Buffalo
vaisshna@buffalo.edu

Ashutosh Shailesh Bhawsar
CSE, University at Buffalo
abhawsar@buffalo.edu

Suraj Suraj
CSE, University at Buffalo
suraj@buffalo.edu

Abstract

Rumors are rife on the web. False claims affect people's perceptions of events and their behavior, sometimes in harmful ways. With the increasing reliance on the Web – social media, in particular – as a source of information and news updates by individuals, news professionals, and automated systems, the potential disruptive impact of rumors is further accentuated. Within NLP research the tasks of stance classification of reddit posts and social media posts and the creation of systems to automatically identify false content are gaining momentum. The proposed models are for completing two of the RumourEval 2019 tasks, the first task is to classify/ detect stances of a tweet's reply thread and the second task is to verify the rumor introduced by the source tweet credibility. In addition to that, a task of generating stance text given a source tweet, prior tweets in the conversation thread and a stance classification label is introduced. We present out approach in the three task vis-à-vis Task A for determining the stance detection of the source texts, Task B where we determine the veracity of the source text. And finally, Task C where build a stance text generator. We use complex models such as BERT, LSTM and seq2seq encoder decoder to achieve desired results.

1 Introduction

The aim of this project is to deal with rumour verification and generation of stance using techniques of NLP. Dataset provided was introduced in RumorEval2019 which contains twitter and reddit posts along with replies. A structure.json file is provided to realize the source and reply structure of conversation threads. The first task is to classify/ detect stances of a tweet's reply thread into 4 classes - Support, Deny, Query, and Comment. The second task is to verify the rumor introduced by the source tweet credibility by classifying source text into 3 classes - True, False, Unverified. The third task is to generate a stance text (R) based on source text (S), replies discussing the claim (CTX), and labels (L) obtained as output from task A by training a language model. The ground truth is the golden text (Y) which will be used for evaluation purposes.

2 Related Work

Our research and experimentation is directed from the previous work done in the several rumor evaluation and veracity detection with several experimentation. RumourEval is a common assignment, coordinated as a component of SemEval in 2017 and 2019, containing two subtasks: (I) Stance classification and (ii) Veracity expectation. In task A, the stance classification is planned as a four-class classification problem, where answers to a social media post (source) can support, deny, inquiry or give a comment to the source. Subtask B comprises anticipating the veracity of talk in social media, based on the text as well as metadata highlights. In Elena Kochelena eta. they group a set of Twitter posts examining tales into by the same token supporting, denying, addressing or commenting on the basic bits of hearsay. They propose a LSTM-based sequential model that, through displaying the conversational construction of tweets, which accomplishes an exactness of 0.784 on the RumourEval test set beating all different frameworks in Subtask A. Ilya Sutskever eta. method utilizes a multilayered Long Short-Term Memory (LSTM) to plan the information sequence to a vector of a proper dimensionality, and afterward one more

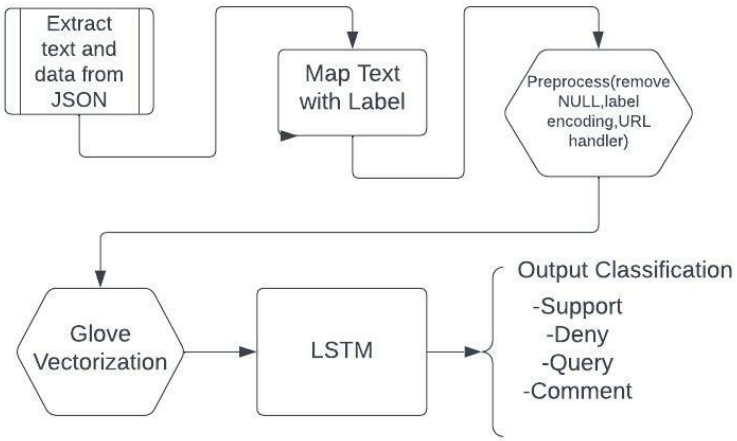

Figure 1: Architecture for LSTM

profound LSTM to disentangle the target sequence from the vector. AndrejJan eta. show that models prepared on English information can be utilized effectively for anticipating stance in different dialects. Thier results feature the potential of multilingual BERT and machine interpretation for rumor analysis in dialects where commented on information is scant or not promptly available.

3 Methods/Model Architecture

3.1 Task A

3.1.1 Preprocessing: Scraping data from JSON files

We are dealing with both source text and reply text, so we extract both texts from JSON files and save them in CSV files for training purposes. For source text, we extracted the text and ID and detect replies if it's a reply to any other text. For each reply, we extract the ID and text along with the source text and its ID with the previous text and ID. If the current text is a reply to a post which is, in turn, a reply to the source post then the current text stance depends on both the previous and the source text. We mapped this data to key.json data based on tweet IDs to get labels for training data. Next, we processed the mentions and URLs in the text by replacing them with $MENTION$ and URL using regex. We performed label encoding for labels as 0,1,2,3 for Support, Deny, Query, and Comment respectively. We saved all data into a single csv file after cleaning the data by removing NULL values and noise.

3.1.2 Models:

For Milestone2 we used MLP with TF-IDF vectorization to classify the inputs. For Milestone3 we used the technique of sentence pair classification using RoBERTA model. Where the input to the model will be the previous replies or the source text and the reply text should be classified as Support, Deny, Query, and Comment. We are using Roberta-large from Roberta tokenizers for the purpose of tokenization. We used class weights in order to deal with imbalanced classes. We tried the LSTM model as text classification with glove embeddings but the RoBERTA model with sentence pair classification outperformed the LSTM model. The structure for sentence pair classification is as follows: Source + Reply1 - label SourceReply1 + Reply2 - label We are combining the source and the previous reply texts as text1 and the reply to be classified as text2 for sentence pair classification.

3.2 Task B

3.2.1 Preprocessing: Scraping data from JSON files

This was the same as that of task A with additional tasks performed as we extracted the source text and its ID from source tweets and mapped them with key labels as the training labels were different than task A. Performed label encoding to convert the classes 'true',' false', and 'unverified' to 0,1,2 respectively.

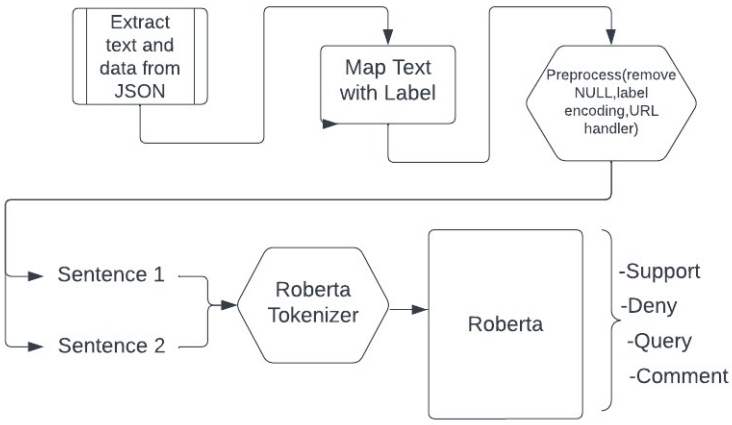

Figure 2: Architecture for sentence pair classification using RoBERTA

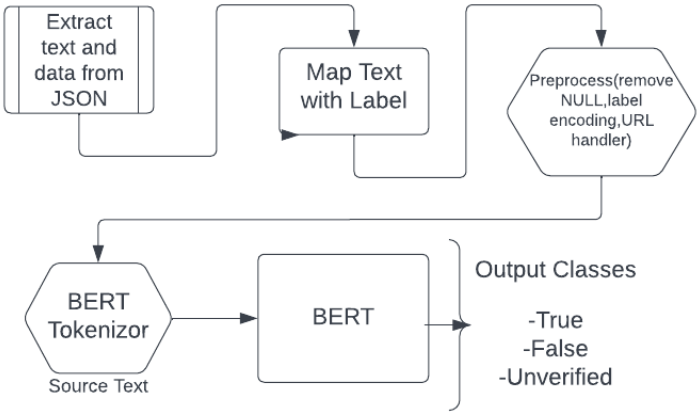

Figure 3: Architecture for LSTM

3.2.2 Models:

For some of the models, we used numerical features like "favourites_count" from Twitter data and "ups" from the Reddit data and categorical features like the labels of task A ,like "Support", "Deny", "Query", and "Comment" as a feature. For the TASK B milestone2 baseline, we used NB, SVM, and MLP with TF-IDF vectorizer with only text data as input. For milestone3 we tried different models like BERT, LSTM, and MLP with numerical data. We tried using the BERT model with text data we could not get a good score. We tried inputting the text numerical features like 'likes' or 'upvotes' by converting them to string sentences. Tried inputting this to the model. For LSTM we used the text data with glove embeddings we got better results compared to BERT as the data size is limited to 365 data points. We tried another approach where we tried using the numerical features like "likes" and categorical features like classes of SDQC of Task A along with the tf-idf vectorized text data to input into the MLP classifier using the pipelining and function transformer. After using the additional features we got the best results.

3.3 Task C

3.3.1 Preprocessing: Scraping data from JSON files

After getting trainingData.csv and testData.csv from preprocessing part of task A and B, we process the data further and get two columns: source text and target text to train our model. We consider source text

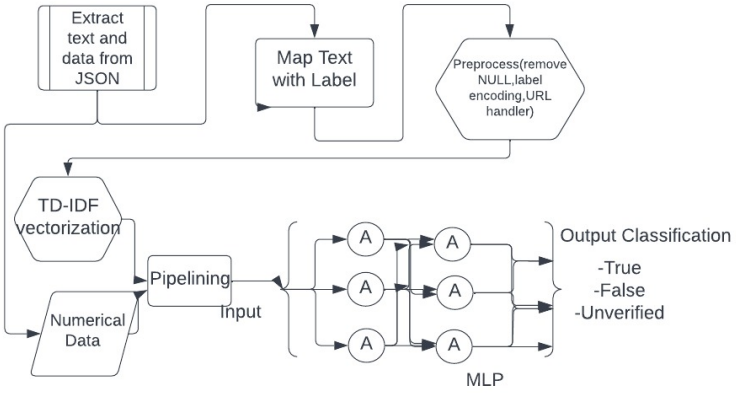

Figure 4: Architecture for combining the tf-idf vectors with numerical data using pipelining

as the cumulative source and level replies. The corresponding target text is the replies to be generated and the concatenation of all the replies in deeper levels of trees. The final data output of preprocessing is saved as taskCTrainingData.csv and taskCTestData.csv.

3.3.2 Models:

For this task, we need to compare the golden text and golden responses. For training, let's consider a conversation where we have one source text and 10 replies to it. We also have a tree like structure with multiple recursive replies to each reply text. Unlike baseline in milestone 2, where we considered a full concatenation of source and all the corresponding replies, we consider the replies only till the 1st level of tree, meaning the replies for original source text and consider a combination of all other replies in a concatenated format. This helps train our model in the given data structure. The main idea is to consider the source text and generate the first reply. Next we consider the source text and 1st reply to generate the 2nd main reply and so on. We also use the source text and 1st reply to generate the lower levels of concatenated replies. This would give us a golden response text. A T5 is a text to text transfer transformer which converts all NLP problems like language translation, summarization, text generation, and question-answering a text-to-text task. It is an encoder-decoder model. For the baseline, we use pretrained simplet5 model, which is further trained for our use case, i.e. input is the text of the source and all replies and output is a generated stance. The model tries to aggregate and summarize the most important information about a given target and a stance. For improving the baseline, we followed the actual structure of given data i.e. the source and replies tree structure and limit to 1 level depth of replies for training considering hardware limitations and maintaining simplicity. Refer figure 5. The blocks represent elements of a sequence and lines represent attention visibility.

Before training, we use some data cleaning strategies by defining some regular expressions and using functionalities from spacy. We introduce the maximum length of source text to 1000 characters and maximum length of generated stance text / target text as 150 characters.

Simple t5 is built on pytorch lightning and huggingface transformers. We use pretrained t5 model and fine-tune it according to our needs. The output of preprocessing is given as the fine-tuning parameters for the simple t5 model. Maximum number of epochs is set to 5.

4 Results

4.1 Task A

Refer figure 6 and 7.

We obtained the best result of 0.6 macro average f1 score using sentence pair classification using RoBERTA.

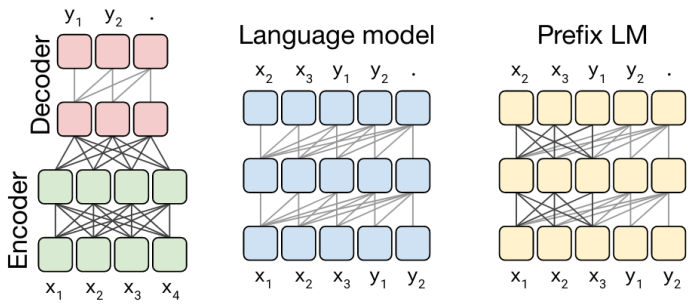

Figure 5: Transformer Architecture

	precision	recall	f1-score	support
0	0.61	0.44	0.51	383
1	0.14	0.10	0.12	88
2	0.19	0.05	0.08	105
3	0.73	0.88	0.80	1032
accuracy			0.68	1608
macro avg	0.42	0.37	0.38	1608
weighted avg	0.63	0.68	0.65	1608

	precision	recall	f1-score	support
support	0.45	0.30	0.36	102
deny	0.37	0.43	0.40	82
query	0.73	0.72	0.73	120
comment	0.89	0.91	0.90	1181
accuracy			0.82	1485
macro avg	0.61	0.59	0.60	1485
weighted avg	0.82	0.82	0.82	1485

Figure 6: Task A: LSTM with glove embeddings (left) and RoBERTA sentence pair classification (right)

4.1.1 Comparison between baseline and this approach:

In the baseline model, we are using TF-IDF with MLP where the text and reply are considered as one input. The macro average f1 score for the baseline is 0.33. In this approach, we used sentence pair classification using transformers in RoBERTA model where the previous text and current text are given as inputs to the model. We were able to achieve a score of 0.6 for the macro average f1 score which is our best result.

4.2 Task B

Refer figure 7 and table 1

4.2.1 Comparison between baseline and this approach:

In the baseline model, we are using TF-IDF with MLP where the text and reply are considered as one input. The macro average f1 score for the baseline is 0.36 and RMSE is 0.571. In this, we tried different models with numerical features as well which boosted our results.

4.3 Task C

Refer figure 8 and 9.
Following are the ROUGE scores calculated for the current model:
'rouge1': AggregateScore(low = Score(precision = 0.4251271271125035, recall = 0.4373630994118866, fmeasure = 0.4161240447328763), mid = Score(precision = 0.441869178964449,

	precision	recall	f1-score	support
0	0.50	0.55	0.52	20
1	0.68	0.72	0.70	32
2	0.53	0.43	0.47	21
accuracy			0.59	73
macro avg	0.57	0.57	0.56	73
weighted avg	0.59	0.59	0.59	73

Figure 7: Task B: MLP with other features

MODEL	Macro average F1	RMSE
BERT	0.26	0.4
LSTM	0.44	0.3
MLP with numeric and categorical features	0.56	0.3

Table 1: Task B Results

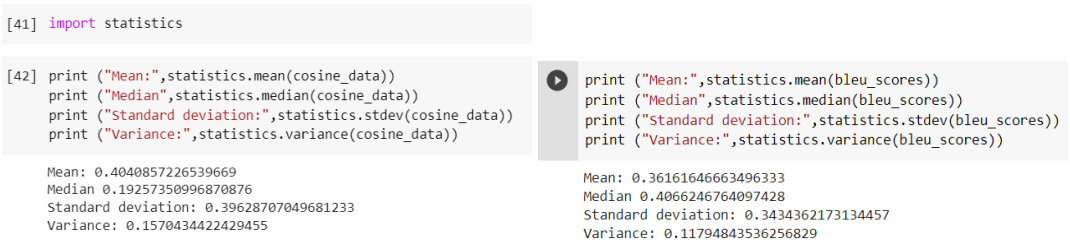

Figure 8: Task C: cosine similarity (left) and BLEU scores (right)

recall = 0.45561177213253007, fmeasure = 0.43370709185583556), high = Score(precision = 0.4586129726488238, recall = 0.47247876916824005, fmeasure = 0.45029000097689753)),

'rouge2': AggregateScore(low = Score(precision = 0.3819559024919208, recall = 0.3999018996642332, fmeasure = 0.3807335663192744), mid = Score(precision = 0.39959570841004133, recall = 0.41828578882066375, fmeasure = 0.39844333431819157), high = Score(precision = 0.41756983963119615, recall = 0.4369868128739215, fmeasure = 0.4163796071175468)),

'rougeL': AggregateScore(low = Score(precision = 0.4164099911290084, recall = 0.4313838334980544, fmeasure = 0.4102650993319388), mid = Score(precision = 0.43291338466662976, recall = 0.449612498737404, fmeasure = 0.42740240715039857), high = Score(precision = 0.4498241332041679, recall = 0.4668836699760964, fmeasure = 0.4446337746276702)),

'rougeLsum': AggregateScore(low = Score(precision = 0.4172381694754507, recall = 0.43122815604303016, fmeasure = 0.4106615766985917), mid = Score(precision = 0.43434342576328455, recall = 0.4503354064217186, fmeasure = 0.42825460863359854), high = Score(precision = 0.45088245743137784, recall = 0.46864159585325754, fmeasure = 0.44556725826435745))

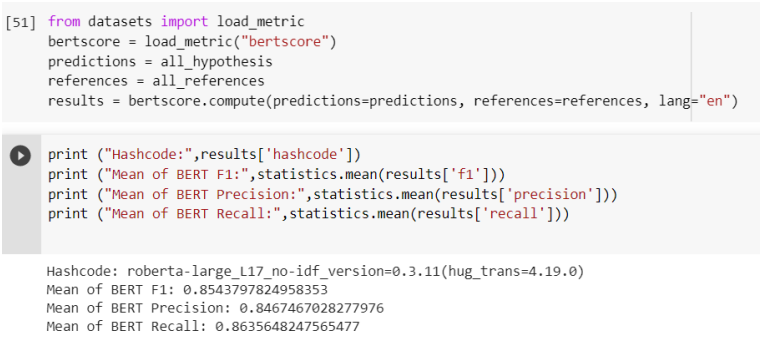

Figure 9: Task C: BERT Scores

```
simplet5-epoch-0-train-loss-1.5848-val-loss-1.3135
simplet5-epoch-1-train-loss-1.2941-val-loss-1.2434
simplet5-epoch-2-train-loss-1.1719-val-loss-1.2004
simplet5-epoch-3-train-loss-1.0613-val-loss-1.1884
simplet5-epoch-4-train-loss-0.9804-val-loss-1.2047
```

Figure 10: Saved Epochs - 5

4.3.1 Comparison between baseline and this approach:

The baseline was using external data 'news_summary.csv' which was initially used for summary generation for news articles. The source text was large news articles and target text was set to the headlines of respective news articles. We used same transformer simplet5 model for the baseline and used concatenated source and all replies as the input to this model.
The new approach doesn't use any external data. Also the input to our encoder-decoder transformer is formatted as it appears in the real conversation - source text, it's replies and all recursive replies to consecutive replies. By this way, even if we are using a pre-trained t5-base model, the whole model is trained on our own data advocating to the original tree structure. The results are significantly better than that of the baseline and all required scores are listed below in Reports section.

5 Discussion and Error Analysis

5.1 Task A

We observed that there is a huge class imbalance in the data for TASK A as there are more than 3000 entries that belong to the comment class whereas other classes have entries less than 1000. The label counts for train data are - comment: 3519, support : 924, query: 395, deny: 378. Most of the misclassified classes are from "Deny". Sentence pair classification gave the best results but the limited data size is a drawback of the data.

5.2 Task B

The size of the data set is small when compared to TASK A. The distribution of training labels is - true : 144, false: 79, unverified: 104. Huge models failed to give the best output as the data size is very small. Addition of other features like "likes" and "task a label" to the machine learning model boosted the score. The deep learning models can be improved by adding a concat_layer with the embeddings of text and the numerical data in this way we can input both the text and the numerical data to the deep learning models.

5.3 Task C

Refer figure 10 and 11. As we can observe, the validation loss normalises (graph flattens) at a higher rate than that of the training loss. The model would be trained to be very specific and would find non existing features and relations between source and target texts if we run more epochs. Hence we limit the number to 5.

6 Conclusion

For Task A we were able to achieve good results by introducing deep learning models , when compared to baseline. For task B due to limited data size machine learning models gave the best results. Fr Task C, since we followed the proper tree structure of conversations and appropriate NLP models, we were able to get decent results. However, using a dual encoder-decoder model and considering all replies of all depths along with labeling results from Task A, we can surely improve this transformer.

7 Work Distribution

For milestone 1, the literature survey for all subtasks was done collectively by Vaisshnavi, Ashutosh and Suraj.

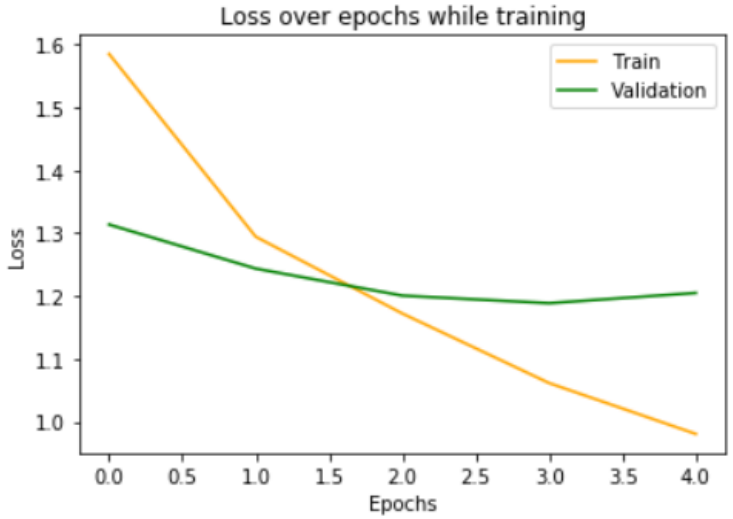

Figure 11: Train and validation loss over epochs

For milestone 2, i.e. for all the baseline models, Task A was handled by Suraj, Task B was handled by Vaisshnavi and Task C was handled by Ashutosh.
For milestone 3, all members of group worked collectively for improving baseline models and achieving good results where Task A was led by Suraj, Task B was led by Vaisshnavi and Task C was led by Ashutosh.

Acknowledgements

Team Predict This! would like to acknowledge the extraordinary debt we owe to Prof. Rohini Srihari for her valuable teachings and lessons for Natural Language Processing concepts which acted as the base for our project. We would also like to acknowledge and give our warmest thanks to Sougata Saha and Souvik Das for the guidance and advice throughout all the milestones for this project. It definitely wouldn't have been possible without their support.

References

https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

Ing. Peter Krejzl. 2018. Stance detection and summarization in social networks. PhD study report Technical Report No. DCSE/TR-2018-01 University of West Bohemia in Pilsen, Czech Republic

https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2018/Rigo_Krejzl_2018-1.pdf

https://pypi.org/project/simplet5/

https://arxiv.org/pdf/1910.10683.pdf

https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks

https://huggingface.co/docs/datasets/v1.0.1/loading_metrics.html

https://jalammar.github.io/illustrated-bert/

https://mccormickml.com/2021/06/29/combining-categorical-numerical-features-with-bert/

https://truthandtrustonline.com/wp-content/uploads/2021/10/TTO2021_paper_31.pdf

https://aclanthology.org/S19-2190.pdf

https://arxiv.org/abs/1704.07221

https://arxiv.org/pdf/1809.06683.pdf

https://arxiv.org/pdf/1409.3215.pdf