Rumour Detection and Stance Classification

Omkar Gurjar, Dhawal Jain, Jatin Paliwal, Mahathi Vempati

November 11, 2019

1 Abstract

Battling the surge of fake news is a multi-faceted, complicated project, a segment of which is to detect whether rumours that spread on social media are fake or real. One way in which this can be done is to analyse the public reaction to a rumour when broadcasted. The public often have information from a lot of other sources, and their collective reaction brings in valuable information in detecting whether a rumour is real or fake. In order to analyse public reaction, we take tweets and Reddit posts on a topic and study all the replies. We classify the replies into support, deny, query and comment (SDQC) and later use this to verify the original post. We have been given tagged data for training for this task.

2 Description of the Dataset

For training purposes, we have been provided with 9 categories about topics that have generated a lot of conversation in Twitter and Reddit - for example, the Charlie Hebdo shooting and the Sydney Seige. Each category contains around 120-150 tweets and their replies in the Twitter tree structure or Reddit comment tree structure and every reply is tagged with Support, Query, Comment, Deny. The test data also contains rumours that need to be classified as true or false. We have with us the training data from the 2017 and 2019 editions of the competition, and the test data from 2019.

The posts themseleves can be considered as trees, with parent and sibling posts providing context. The tagged data contains whether a given tweet is S, D, Q or C with respect to its immediate ancestor.

- 2019 Train and Test Data
- 2017 Train Data

3 Related Work

The first part of our problem - classifying whether a reply to a post is Support, Deny, Query or Comment- is a standard NLP problem known as stance classification, and there are several papers on various approaches for the same. Separately, rumour detection is also an ongoing field of research in NLP. Kuntal Dey et al give a simple SVM based model for the classification problem and achieve an F-score of 74.44 on the 2016 version of the semeval data, beating the then state of art. Michal Lukasik et al attempt a three class classification (support, deny, query) on social media posts, and then use it to detect rumourous tweets from the 2011 England riots. They use Gaussian processes for classification and achieved around 80% accuracy on their dataset. Ahmet Aker et al. try out simple classifiers like decision trees and random forests. This paper focuses on feature extraction from the tree structure of social media data - for instance, text similarity between original post and replies, before feeding it into a support vector classifier.

4 Subtask A

This subtask was completed in the first phase of the project. We write the details here for

completeness: It deals with tracking how other sources orient to the accuracy of the report in question (that we need to eventually find the veracity of). We are provided with a tree-structured conversation formed of posts replying to the original rumourous post, where each post presents its own type of support with regard to the rumour. We frame this in terms of supporting, denying, querying or commenting on (SDQC) the claim. The goal is to label the type of interaction between a given statement and a reply post.

4.1 Baseline Methodologies Attempted

Repository: Transformers Repository

To begin with, we attempted a naive context free method - classify whether a reply is S,D, Q, C only with the text features, disregarding even the original post. Note that this attempt is substantiated as a tweet by itself could give reasonable information about its class. A question mark or a word like 'how', etc, immediately hints at a query, while negative connotations could refer to a denial.

For this approach, we use Transformers: an attention mechanism that learns contextual relations between words (or sub-words) in a text as opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that its non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

We use SimpleTransformer: This library is based on the Pytorch Transformers library by HuggingFace. Using this library, you can quickly train and evaluate several transformer models. We first preprocess the data and convert the JSON tree structure to Tab separated values fed in a 2D array to the SimpleTransformer module.

The transformer then has two phases:

- The pre-train phase: which has already been done and is available - this is the phase where the transformer learns the structure of a language after being trained on a large corpus (eg: Wikipedia).

- The fine-tune phase: This is separate for every transformer, this is where we feed in our tagged data to the transformer and it learns the S, D, Q, C tags. For each of the 3 transformer models we used, fine tuning took about 5-6 hours.

We attempt three different transformer models here:

4.1.1 Bert

Details 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on cased English text.The pre-training corpus for BERT is BooksCorpus (800M words) and English Wikipedia (2,500M words)

Accuracy: 70.71 %

4.1.2 Roberta

Details 125M parameters RoBERTa using the BERT-base architecture. RoBERTa uses 160 GB of text for pre-training, including 16GB of Books Corpus and English Wikipedia used in BERT. The additional data included CommonCrawl News dataset (63 million articles, 76 GB), Web text corpus (38 GB) and stories from Common Crawl (31 GB).

RoBERTa builds on BERTs language masking strategy, wherein the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples.

RoBERTa, which was implemented in PyTorch, modifies key hyperparameters in BERT, including removing BERTs next-sentence pre-training objective, and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance.

Accuracy: 73.74 %

4.1.3 XLNet

Details 12-layer, 768-hidden, 12-heads, 110M parameters. XLNet was trained with over 130 GB of textual data.

XLNet is a generalized autoregressive pretraining method. It is bidirectional and heavily relies on the pretraining phase for language structure compared to the other two models using a method called Permutation Language Modelling.

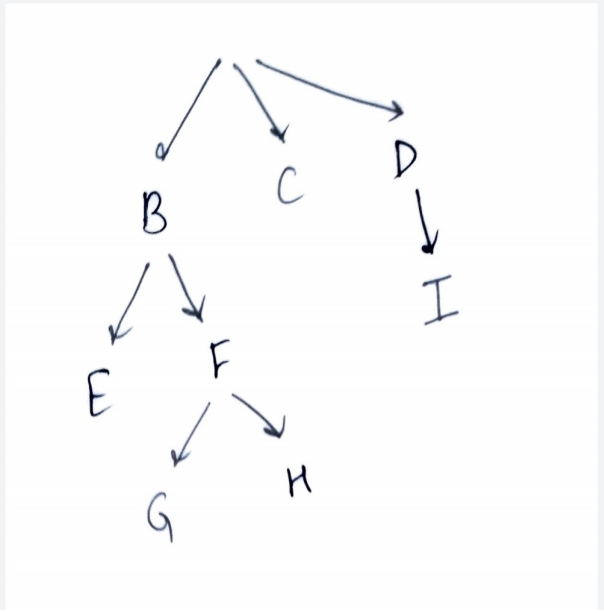
Accuracy: 77.6 %

4.2 Architecture of Final method

Repository: branchLSTM Repository

Given the poor performance of our context-free models (note that a naive classifier that predicted 'Comment' – the majority class – all the time on the test set would have given an accuracy of 75.58 %), we move on to a contextual model.

The structure of the data in our project is in a tree format. Consider the example in the figure below.



Consider three starting tweets: B, C and D. All other tweets are replies to these. Now, for all the tweets in the sub-branches B, C and D, all the tweets make up the context. For example, for tweet G - tweets B, E, F and H make up the context. However, one way to look at this data, as we need to do stance classification, is to ignore the context due to siblings and look at only parental context.

Therefore, the context for G is B $\rightarrow$ F $\rightarrow$ G. The entire tweet tree is converted to such branches. For example, if the tweet tree is as above, then the branches are:

$$B \rightarrow E$$

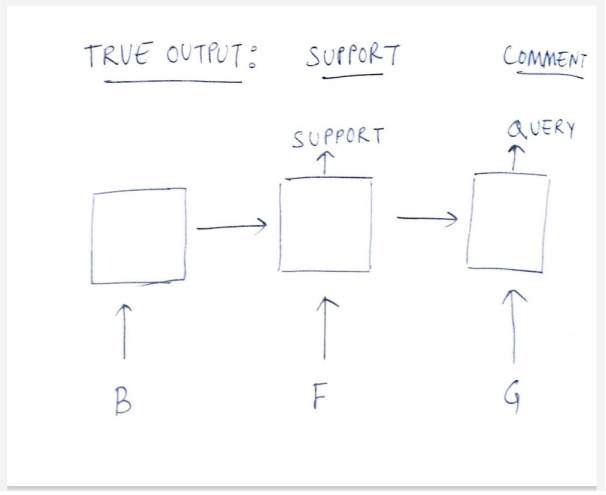
$$B \rightarrow F \rightarrow G$$

$$B \rightarrow F \rightarrow H$$

$$C$$

$$D \rightarrow I$$

Now, this data is an extremely convenient format to input into a Recurrent Neural Network as every post is tightly correlated to its immediate ancestor, but also needs to learn context from other posts in the branch. Consider the branch $B \rightarrow F \rightarrow G$:



Now, F has a tag on its stance with respect to B, and G has a tag with its stance with respect to F. Say, F is tagged support and G is tagged comment. The branch is given as input in parts at various timesteps, and the loss function can be calculated based on every output.

Since RNNs are known for forgetting historical context, we can use an LSTM to get past this problem. Hence, if we do not care about

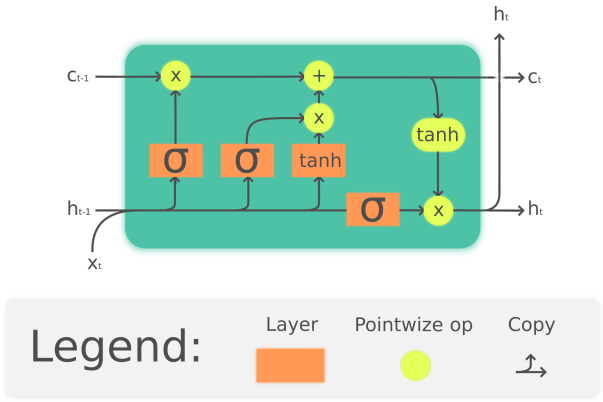


Figure 1: the LSTM unit cell - each node has learnable parameters for a forget and retain gate - to ensure important information from way back in the branch is retained.

sibling context, a branch-LSTM is an ideal neural network to use.

The Lasagne module constructs a suitable LSTM given input hyperparameters. We did not do a hyperparameter search but used the optimal values from the research paper in the repository. Once converted to branch structure, it is straightforward to feed it into the LSTM and train the model. Then, the test data is also converted to branches and tagged. We converted the data to the 2017 contest format, and the links to the data are available in the repository - which is the baseline code modified.

4.3 Evaluation Mechanism and results

For evaluation, we had the test data set as well as a results analysis function that was provided - which provided an accuracy, F-score and confusion matrix. On running the model on the 2019 (Tweets only) data, the results are as follows:

Accuracy: 95.77 %

F-score: 0.756

4.4 Analysis

This heavily beats the 84% accuracy of the winning model in SemEval. While the two cannot be compared because we have only trained and

```
Accuracy = 0.957786116323

Macro-average:
Precision   0.821
Recall      0.724
F-score     0.756
Support     --

Per-class:
            Comment     Deny        Query       Support
Precision   0.978       0.750       0.784       0.773
Recall      0.981       0.375       0.690       0.850
F-score     0.980       0.500       0.734       0.810
Support     956         8           42          60
```

Figure 2: Precision, recall and f-score

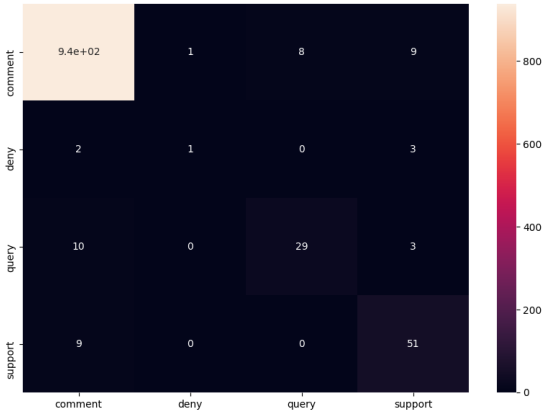
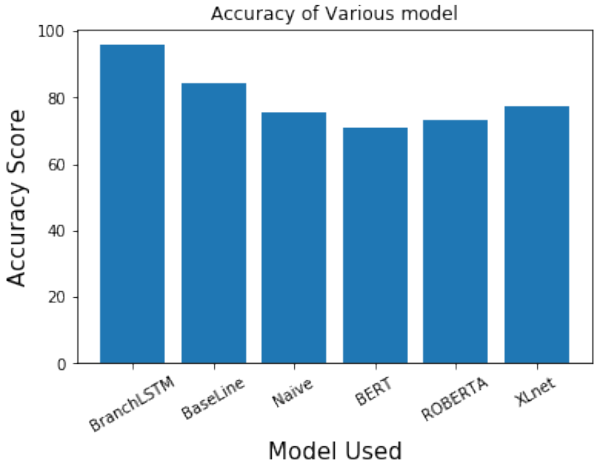


Figure 3: Confusion Matrix



tested on Twitter data as opposed to both Twitter and Reddit done by the winning submission

We have also attempted to use Bert again for contextual classification - using bigram sentences of ONLY a tweet and it's ancestor, however this also produced poor results with an accuracy of around 77%. It is possible that bidirectionality, while is generally an advantage, is not so in this case as directionality is important, a tweet's stance is given with respect to its parent, and the other way around could provide spurious results.

5 Subtask B

The goal of the second subtask is to predict the veracity of a given rumour. The rumour is presented as a post reporting or querying a claim but deemed unsubstantiated at the time of release. Note that for each of these posts we also have the public opinion we have analysed in Task A. Given such a claim, the system should return a label describing the veracity of the rumour as true or false along with a confidence score.

Based on the rumour, we have to divide it into three classes - True (Class 0), False (Class 1) and Unverified (Class 2). Unlike the previous data, which had sequential dependence, the data here is in the form of a bag of words representing the tweet or the Reddit post itself and three more features comprising the support ratio, deny ratio and comment ratio from Task A (The query ratio is redundant.)

5.1 Baseline Methodologies

We use the Baseline code provided in the competition which essentially uses all features - A bag of words of the tweet itself, and the three ratios, and runs a linear classifier on it. We attempt to change the feature-set and the classifier types.

5.2 Final Model Architectures

The models we have used are: a linear classifier, MLP classifier, Logistic regression classifier, Decision Tree and Random forest. These 5 architectures have been tested on two different feature sets (1) The complete set, (2) Only the support ratio, deny ratio and the comment ratio.

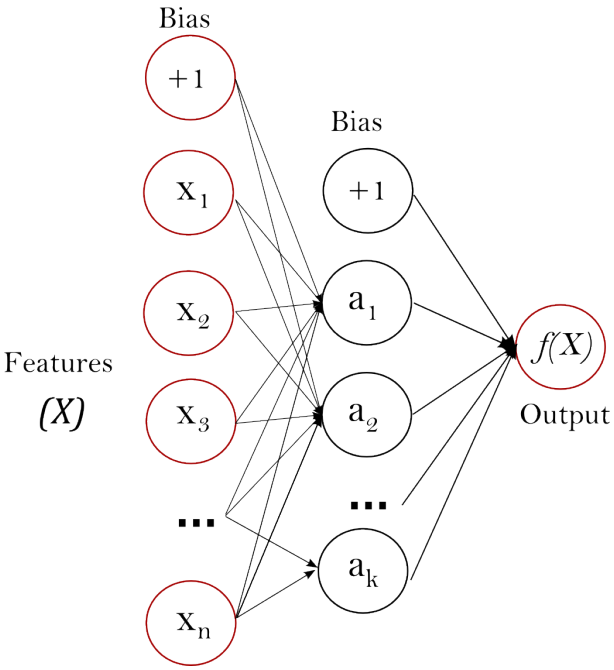


Figure 4: An MLP with just 3 inputs gave us the best features. We used 100 neurons in the first layer and 50 neurons in the second layer.

5.3 Evaluation Mechanism and Results

For evaluation, once again, we have accuracies and F1-scores after comparing with the correctly tagged test data. The F1-scores, confusion matrices and a few comparison graphs are given below. For the confusion matrices, the correct classifications are on the y axes, and the predicted classifications are on the x axis. For a good result, the diagonal has to have maximum values (should be lightest according to the used colour scheme.) Thus, we can compare the different results in the images given below.

Note that we have ten different trials, 5 using the full feature set, and 5 using only the three ra-

tios, and each of the 5 represent different classifiers.

The best accuracy using all the post features was 53.08 % with a Random forest, whereas the best accuracy using just public reaction is 54.32 % with an MLP Classifier. It turns out the accuracy that was gotten by using 1750+ features could be surpassed by the three features describing public reaction.

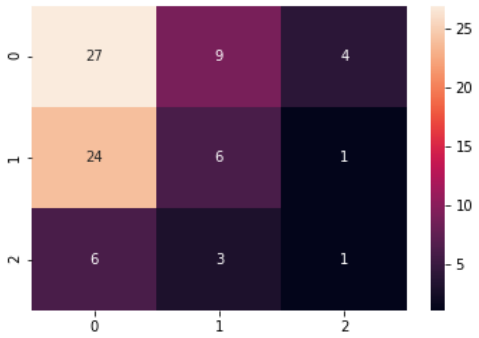


Figure 5: Linear SVC - Full feature set

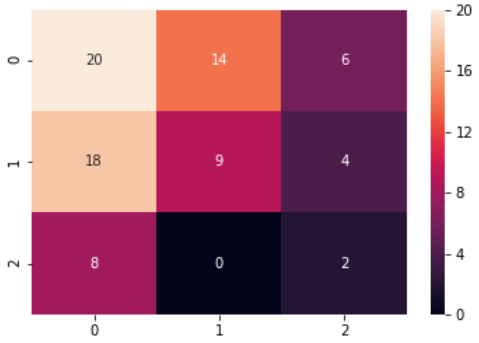


Figure 6: MLP Classifier - Full feature set

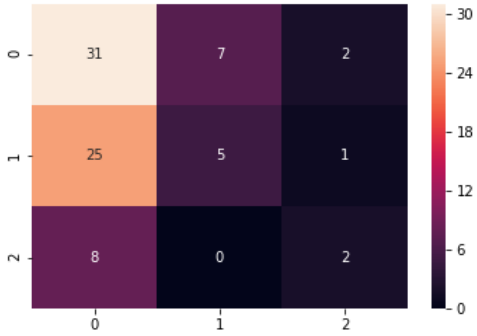


Figure 7: Logistic Regression - Full feature set

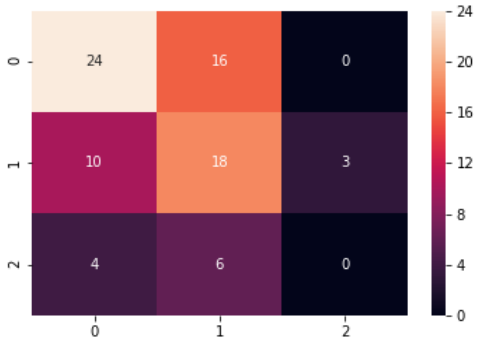


Figure 8: Decision Tree - Full feature set

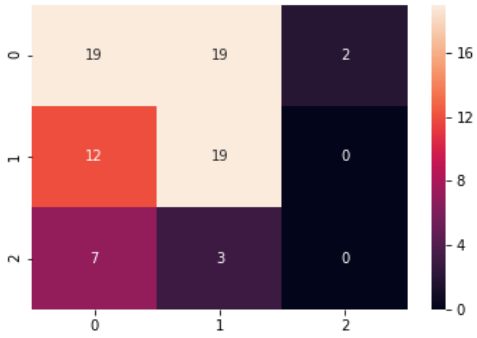


Figure 9: Random Forest Classifier - Full Feature set

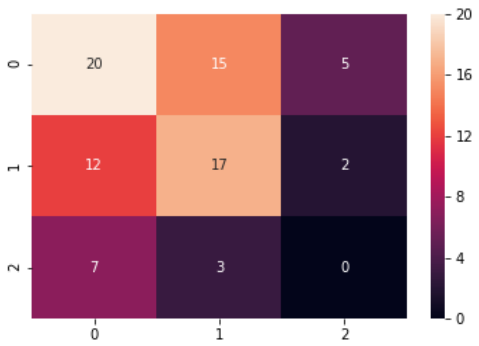


Figure 10: Linear SVC - Only public reaction

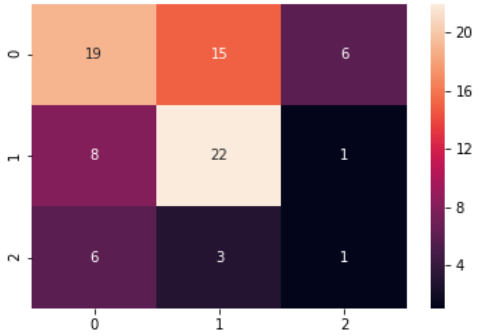


Figure 11: MLP Classifier - Only public reaction

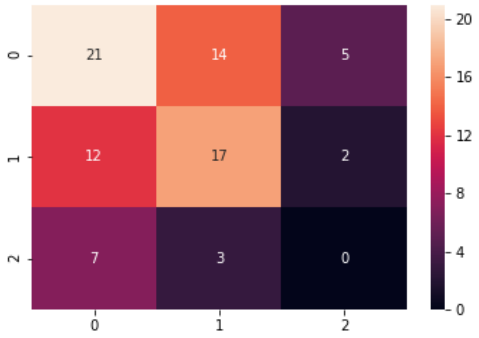


Figure 12: Logistic Regression - Only public reaction

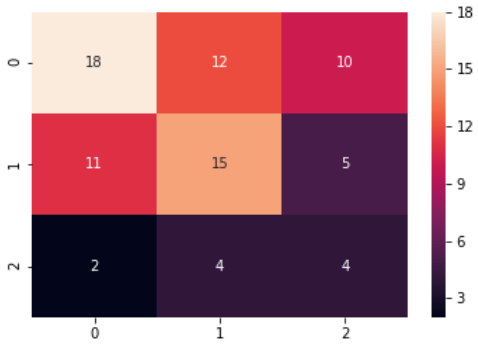


Figure 13: Decision Tree - Only public reaction

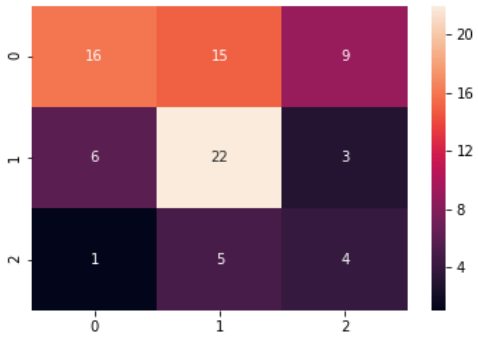


Figure 14: Random Forest Classifier - Only public reaction

5.4 Analysis

When we analyse the confidence scores in each of the ten cases, we notice something interesting. The best classifier on the 1750 feature vector was the random forest, which had a **confidence score** of around 0.63 whereas the MLP, which was the best classifier for the 3-feature input gave a confidence of 0.71. Therefore the starkly lower number of features not only gave a better result, but also gave the result more confidently, making it clear that the support, deny and comment ratio were the more significant features needed in this task.

Here, we have the Accuracies and F1 scores for the various models below, as well as the average confidence scores for the 3-feature set.

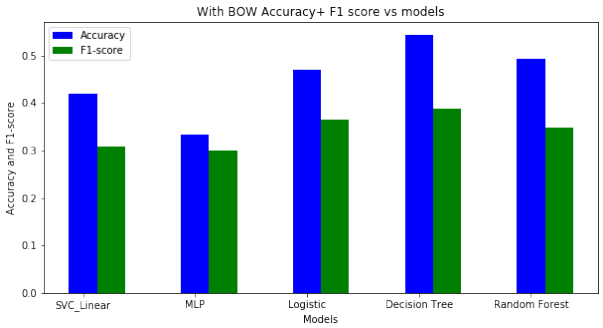


Figure 15: Model Comparison - Full feature set

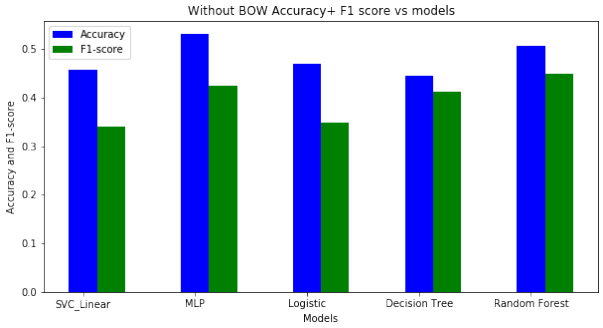


Figure 16: Model Comparison - only public reaction

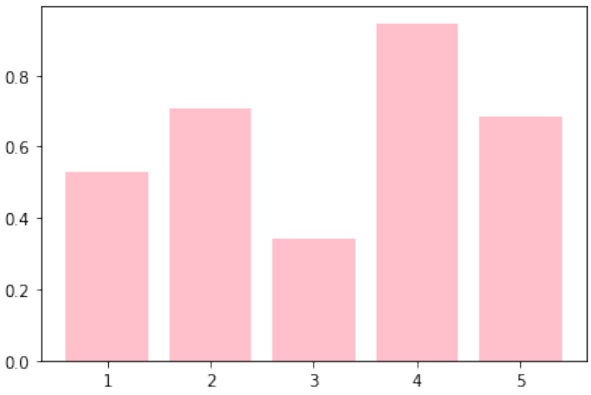


Figure 17: Confidence scores - Three-feature trials for various models. The models are (0) Linear SVM, (1) MLP, (2) Logistic Regression, (3) Decision Tree, (4) Random Forest

6 Conclusion

We got extremely satisfactory results for task A, and we did do much better than baseline in Task B. The scope for improvement in Task B lies in the fact that it is difficult to have just a tweet and its replies and perform rumour classification - a much better accuracy could possibly be achieved if external context from sources such as Wikipedia and news articles were also used as part of the feature set for classification.

7 Links

- Video
- Website
- Github Repo for Transformers
- Github Repo for branchLSTM and subtask B