IR Term Project-Rumour Detection

TEAM ID :-18

Harshavardhan Alimi	-	18EC10021
Nalla Siva Krishna Sai	-	18EC10037
Vunnam Yashwanth Babu	-	18EC10069
Ravi Karthik	-	18EC3AI19

Individual Contributions:-

Common task(Task-1) :- Done by Harshavardhan Alimi, Ravi Karthik, both have done the task while on the call.

Contribution:-
- Generated the corpus file by following the steps mentioned in the Readme file,It took a lot of time as the Dataset is very large.
- Merged the available data on stance-labels into 1 file and used it for labelling tweets, as each source tweet is represented in tree-format,we have added stance(key) for each tweet in the dictionary, stance is the one-hot encode of stance-label:= [0, 0, 0, 0,]:- No stance ; [1, 0, 0, 0]:-comment ; [0, 1, 0, 0]:-deny ; [0, 0, 1, 0]:-support ; [0, 0, 0, 1]:-query. Lot of tweets doesn't have stance label, we managed them giving separate encode label([0,0,0,0])
- Read the paper and Run the Tree-LSTM code for the tree dictionary to generate a Rumour label for source-tweet.
- Read the paper and Run the code for Multi-Task paper which uses branch-LSTM architecture.

Planned:-
- The Tree-LSTM will find hidden representations for each node in tree(i.e. Each tweet)[which is already implemented],they can be passed into a feed-forward dense layer and then softmax layer to give 4 outputs from there we can get the stance for each node.

Rumour_RVNN(Task-2) :- Done by Nalla Siva Krishna Sai, Vunnam Yashwanth Babu, both have done the task while on the call.

Contribution:-
- Read the paper and run the code in 'Rumor Detection on Twitter with Tree-structured Recursive Neural Networks' paper on their dataset for a few epochs. The data set was very large so had run only few epochs
- Understood the columns used in the processed data set used in the paper
- Partially pre-processed the PHEME dataset , created a vocabulary , and assigned indices to the words in the tweets. Assigned indices to every tweet id and also mentioned the index of the parent of the tweet id as required.

Planned :-

- Using the created vocabulary , we have to include index:count for each word in each tweet id and also we have to mention the maximum length of all texts under a single root id for all tweet ids in that tree
- Create another text file for the labels for every tweet id
-

Results:-

Verified-Summarization paper :-

1. Stance-labels for sub-set in each dataset:- (remaining tweets in the respective dataset doesn't have stance labels)

Dataset	comment	deny	support	query
charliehebdo	777	57	61	248
ebola-essien	21	6	1	6
ferguson	759	88	108	187
germanwings	229	15	43	79
gurlitt	0	0	0	0
ottawashotting	561	78	82	169
price-toronto	64	7	11	21
putinmissing	33	6	5	18
sydneysiege	760	89	110	225

2. Run the code for the paper on PHEME9-dataset(not including stance-labels at Tree-lstm)

Test-Dataset	Loss at last Iteration	accuracy	f1-score
charliehebdo	45.36610031	0.7552657974	0.639487599
ebola-essien	62.90629578	0.07142857143	0.06666666667
ferguson	51.95731735	0.2569444444	0.2078257156
germanwings	61.10250092	0.5210918114	0.4896692453
gurlitt	63.82353973	0.6666666667	0.5341614907
ottawashotting	52.81412888	0.4941451991	0.3934220305
price-toronto	59.99810028	0.07518796992	0.07330198833
putinmissing	62.06360626	0.4222222222	0.3072368421
sydneysiege	50.19621658	0.5760683761	0.3764398446

Multi-Task for veracity :-

1. Run the code for pipeline containing - Veracity + stance on RumEval dataset

best loss: 1.1639434080521696
Params: {'batchsize': 32, 'l2reg': 0.001, 'learn_rate': 0.0001, 'num_dense_layers': 2,
'num_dense_units': 300, 'num_epochs': 10, 'num_lstm_layers': 2, 'num_lstm_units': 100},

'TaskA': {
        'accuracy': 0.6840077071290944,
        'Macro': {
                'Macro_Precision': 0.3417009142979913,
                'Macro_Recall': 0.33086565906878984,
                'Macro_F_score': 0.31987530722203256},
        'Micro': {
                'Micro_Precision': 0.6840077071290944,
                'Micro_Recall': 0.6840077071290944,
                'Micro_F_score': 0.6840077071290944},
        'Per_class': {
                'Pclass_Precision': array([0.28181818, 0.76834862, 0.1627907 , 0.15384615]),
                'Pclass_Recall': array([0.33333333, 0.87126138, 0.1   , 0.01886792]),
                'Pclass_F_score': array([0.30541872, 0.81657526, 0.12389381, 0.03361345])}},

'TaskB': {
        'accuracy': 0.48148148148148145,
        'Macro': {
                'Macro_Precision': 0.501994301994302,
                'Macro_Recall': 0.5075757575757576,
                'Macro_F_score': 0.4723389355742296},
        'Micro': {
                'Micro_Precision': 0.48148148148148145,
                'Micro_Recall': 0.48148148148148145,
                'Micro_F_score': 0.48148148148148145},
        'Per_class': {
                'Pclass_Precision': array([0.46153846, 0.6   , 0.44444444]),
                'Pclass_Recall': array([0.75   , 0.27272727, 0.5   ]),
                'Pclass_F_score': array([0.57142857, 0.375   , 0.47058824])}}

Rumour_RVNN:-

```
(base) C:\Users\user\Desktop\Rumor_RvNN-master\model>python Main_BU_RvNN.py
WARNING (theano.tensor.blas): Using NumPy C-API based implementation for BLAS functions.
loading tree label
1490
reading tree
tree no: 1793
loading train set
180 160 176 173
loading test set
40 45 44 39
train no: 689 689 689 689
test no: 168 168 168 168
dim1 for 0: 2 79 79
case 0: [ 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
 -1 -1 -1 -1 77] [8. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0.] [    0 2339  168   14 1327   16  724  278  317    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0]
C:\Users\user\Desktop\Rumor_RvNN-master\model\BU_RvNN.py:226: UserWarning: DEPRECATION: If x is a vector, Softmax will not automaticall
y pad x anymore in next releases. If you need it, please do it manually. The vector case is gonna be supported soon and the output will
 be a vector.
  return T.nnet.softmax( self.W_out.dot(final_state)+ self.b_out )
Recursive model established, 1.684862446784973
epoch=0: loss=0.844494
2021-10-13 15:20:16: Loss after num_examples_seen=689 epoch=0: 0.844494
results: ['acc:', 0.381, 'Favg:', 0.33737, 0.9104, 0.4544, 'C1:', 0.75, 0.3333, 0.05, 0.087, 'c2:', 0.7202, 0.4615, 0.2667, 0.338, 'c3:
', 0.7381, 0.5, 0.4773, 0.4884, 'c4:', 0.5536, 0.3085, 0.7436, 0.4361]
epoch=1: loss=0.723861
epoch=2: loss=0.628522
epoch=3: loss=0.560067
epoch=4: loss=0.511090
epoch=5: loss=0.453832
2021-10-13 16:10:48: Loss after num_examples_seen=4134 epoch=5: 0.453832
results: ['acc:', 0.5298, 'Favg:', 0.51335, 0.8681, 0.4311, 'C1:', 0.6726, 0.4, 0.75, 0.5217, 'c2:', 0.7738, 0.6842, 0.2889, 0.4063, 'c
3:', 0.8214, 0.64, 0.7273, 0.6809, 'c4:', 0.7917, 0.5833, 0.359, 0.4445]
epoch=6: loss=0.419242
epoch=7: loss=0.388257
epoch=8: loss=0.356824
epoch=9: loss=0.310192
epoch=10: loss=0.294788
2021-10-13 16:52:10: Loss after num_examples_seen=7579 epoch=10: 0.294788
results: ['acc:', 0.5595, 'Favg:', 0.56117, 0.829, 0.4123, 'C1:', 0.7381, 0.463, 0.625, 0.5319, 'c2:', 0.7738, 0.6, 0.4667, 0.525, 'c3:
', 0.8571, 0.7778, 0.6364, 0.7, 'c4:', 0.75, 0.4651, 0.5128, 0.4878]
epoch=11: loss=0.274244
epoch=12: loss=0.249457
epoch=13: loss=0.232113
epoch=14: loss=0.209367
```