Group Project submitted for course --
Introduction to Deep Learning with Python

Project Title:
Fake News Detection

Group Member:
LI WANSHUI 1530005017
XIE WEIYAN 1530005037
YU ZHEWEN 1530005047

December, 2018

A. Problem Definition

1. Project Overview

With the development of social media, such as Twitter, Facebook, Weibo and so on, there are more available outlets for the society to obtain the current affairs. Differing from the traditional media platforms, the spread of news in these new social media is much quicker and easier. At the same time, there are more and more fake news in these media platforms. A most significant problem is that some false news that are eye-catching to increase the number of users to a website are produced.

In order to decrease such instances and rate various websites for being trustworthy, we have an idea to build the model that can identify the relationship between headline and article.

In this project, we created a model that finds the similarity between the Headline and the article Body and classifies it into 4 classes: unrelated, discuss, agree and disagree. The model uses Bi-directional LSTM for summarizing the sequence and CNN for classifying the summarized sequence.

2. Problem Statement

To state the problem formally, the model is given a headline and an article corresponding to it. The model will find the probability of the pair belonging to each of the Stance class (unrelated, discuss, agree and disagree).

Input:

A headline and a body text

Output:

Classify the stance of the body text relative to the claim made in the headline into one of four categories:

Agrees: The body text agrees with the headline;

Disagrees: The body text disagrees with the headline;

Discusses: The body text discusses the same topic as the headline, but does not take a position;

Unrelated: The body text discusses a different topic than the headline.

B. Data Description

1. Overview of Dataset

The dataset for this problem is collected from the fakenewschallenge.org . The dataset contains the headline, article pair and their corresponding relationship: unrelated, discuss, agree, disagree. There are totally 49972 training samples. The training data is divided into two files:

a. Train_stances.csv : It contains the headline, Body_ID and stance
b. Train_body.csv : It contains the Body_ID and the Body text

Testing data has 25413 unlabeled samples.

Following is one of the row in Training_stance.csv:

"Police find mass graves with at least '15 bodies' near Mexico town where 43 students disappeared after police clash,712, unrelated"

The first field is the Headline, second is the Body_ID and the third field is the stance.

Following is one of the row in Training_body.csv

"
0,"A small meteorite crashed into a wooded area in Nicaragua's capital of Managua overnight, the government said Sunday. Residents reported hearing a mysterious boom that left a 16-foot deep crater near the city's airport, the Associated Press reports.
"

The first field is the Body_ID, second is the body text

So, the first thing which is supposed to be done is to join the two files on Body_ID.

2. Data Distribution

The distribution of the data among the 4 classes is as follows:

We can clearly see that the data is imbalanced

	Unrelated	Discuss	Agree	Disagree
Percentage of training data	73.13	17.82	7.36	1.68

Following are the statistics for the number of words (including punctuations) in Headline and Article

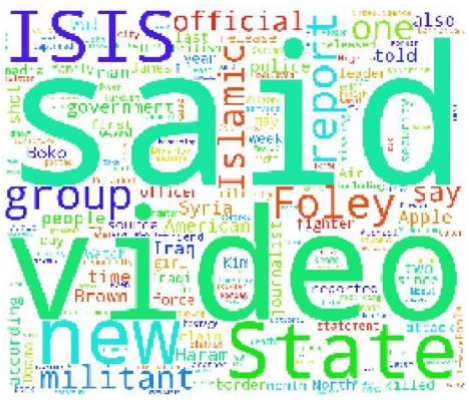
1. The average Headline length is 11.12 and average article length is 369.69
2. The max Headline length is 40 and max article length is 4788
3. The min Headline length is 2 and min article length is 4
4. The median Headline length is 10 and median article length is 304

3. Data Visualization

To find the most important words that occur in each Headline-Article pair corresponding to each class, I concatenated the Headline and Article and made the word-cloud for each of the class

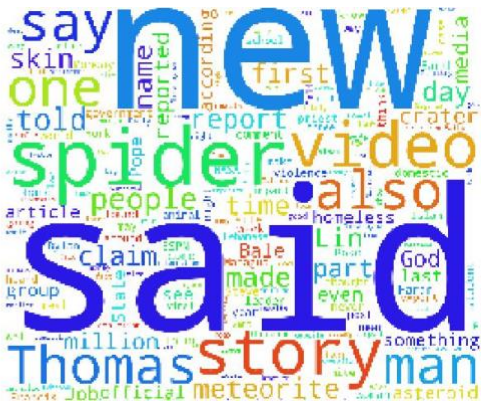
For the unrelated class

For the discuss class



For the agree class

For the disagree class



Obviously, we can see that there is not much difference between the words used in different classes. On a whole, they are pretty much the same which is evident from the Word Cloud. This is the reason why using bag of words is a bad strategy for Fake news detection. More important is to understand and summarize the headline-article pair.

C. Algorithm and Technique

1. Word Vectors

Words cannot be used directly for training. They must be converted in a form that is understandable by the machine learning algorithm. Therefore, we represented words in the form of word vector and used the Global Word Vector (GloVe) of dimension 50 provided by Stanford University. GloVe is an unsupervised learning algorithm for obtaining vector representations for words.

Word Vectors are vector representation of the words and they are obtained by training neural networks on large corpus of global words. They represent words as multidimensional continuous floating numbers where semantically similar words are mapped to proximate points in geometric space. In simpler terms, a word vector is a row of real valued numbers where each point captures a dimension of the word’ s meaning and where semantically similar words have similar vectors.

Here is an example about how a word is represented by its word vector:

Economic

[-0.00023945 0.39277 -0.16155 -0.034783 -0.069467 -0.51534 0.19859 -0.93728 -0.10969 0.098098 -0.068656 0.016178 -0.72234 0.29057 -0.097483 0.31853 0.45386 -0.85754 0.73634 0.1822 0.37787 -0.67011 -0.67249 -1.1899 0.52358 -1.3483 -0.10983 -0.13769 0.014085 1.6902 4.1452 0.87261 0.35469 -0.90482 -1.0903 -0.59752 -1.4137 0.194 0.12474 -0.0775 -1.9835 -0.17015 0.53512 -0.67014 0.27473 0.18497 0.099552 1.2488 0.83231 0.020399]

These are the steps we converted the headline and article to their corresponding word vector form:

1. For each character in the text, check if it is alphanumeric or is a space. If yes, then append it to a list. If not, then append a space. At the end we will have a cleaned string of the text. This is done for all the headline and Article text;
2. Split the headline and Article on spaces;
3. Sum the word vectors of all the headline words (in case they are present in the GloVe dictionary), naming it as titleVec;
4. Sum the word vectors of all the Article words (in case they are present in the wordVector dictionary) , naming it as articleVec;

   *Remark: for the words that do not exist in the GloVe dictionary, we omitted them. As most of words used in news frequently have already included in GloVe dictionary, we considered those words that do not exist in the dictionary as the unimportant contents reflect the main idea of the headlines and body articles.

5. Make another vector, diffVec as bodyVec – titleVec;
6. Normalize the diffVec and it becomes the sample representation.

The reason why we summed up all the all the word vectors and not simply append them is because the dimensions were becoming very large and it needed much powerful GPU for the computation. Also, article length ranges from 4 to 4788 words. Definitely some limit had to be specified to constraint the number of word vectors to be appended.

Definitely, this would result in loss of information about the word ordering in the headline and in the text. However, considering the computational cost, we believe this is a valid constraint.

After converting to the word vectors, we used the Bi-directional LSTM to conduct the headline-article pair text summarization and used CNN for classification.

2. Bi-directional LSTM for Text Summarization

Bi-directional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. We propose a bidirectional model that has the ability to model both the history textual context and the future one to generate multi-sentence summaries. The source sequences are reversed where the output of the forward encoder is fed into to the backward decoder and the output of the backward encoder is fed into the forward decoder.

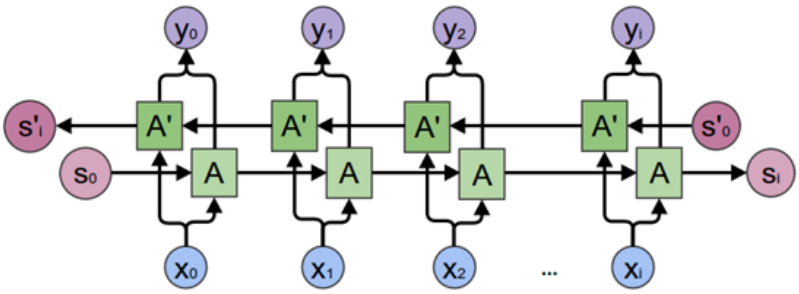


Figure: the structure of Bi-directional LSTM

In this work, the role of the Bi-directional LSTM could be seen as two separate LSTM: one decodes the information from left-to-right, called forward decoder, while the other decodes from right-to-left, called backward decoder.

According to some reports, the performance of RNN-based encoder-decoder models is quite good for short input and output sequences; however, for longer documents and summaries, these models often struggle with serious problems such as repetition, unreadability, and incoherence. Moreover, they tend to generate unbalanced output.

Our proposed model is different from the related work in the sense that it utilizes both the long-term history and future context by using the bidirectional encoder-decoder architecture.

3. CNN for classification

As we known, CNN is great at classifying data, so we choose the CNN to do the classification after the processing in the Bi-directional LSTM. The input to the CNN are vectors of dimensions of 50*(number of Bi-LSTM units). Let' s say that there are 64 Bi-LSTM units, then the input to the CNN would be a final vector of dimension 50*128. (Remark: for the Bi-directional LSTM, the information is processed from both sides, so the dimension of output for Bi-directional LSTM will be 50*128 if we have 64 Bi-LSTM units.)

After a series of convolutional operations and max pooling layers, the model outputs the classification probability for each class.

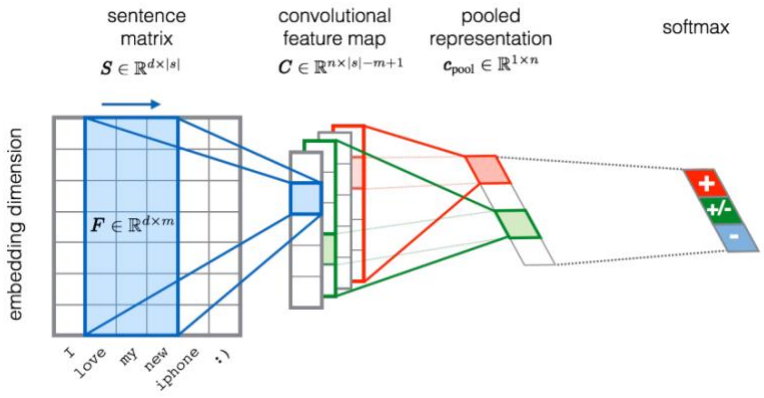


Figure: CNN in Text Classification

4. Addressing with the Imbalance

As shown in the data distribution before, the dataset is imbalanced. While over 70% of news are unrelated, only less than 2% of news are disagreed.

Obviously, the imbalance in the dataset will cause some problems. It is mainly because in most machine learning algorithms, the distribution of classes in new, unlabeled data is assumed as same as that in the training data. The problems exist seriously when the training data does not have the same distribution as the unknown data that needs to be classified.

In order to address these problems, we decided to conduct cost sensitive technique which addresses the learning itself and keep the original dataset completed. The main idea of this technique is to give higher penalty to the network when it misclassifies the minority classes (disagreed news in our case) during training, thus encouraging it to learn those classes better.

Therefore, in our model, we have specified a class weight which is inversely proportional to its percentage in the training data. The weights updates according to those proportion so that the model heavily penalizes if it wrongly classifies disagree but there' s a small penalty when it wrongly classifies the unrelated class.

5. Optimizer and Activation Units

We selected Adam as the optimizer as it is one of the most popular optimizer available and has proven to give good results. Also, it has 2nd order momentum and not just 1st order which again makes it one of the best optimizer for our case.

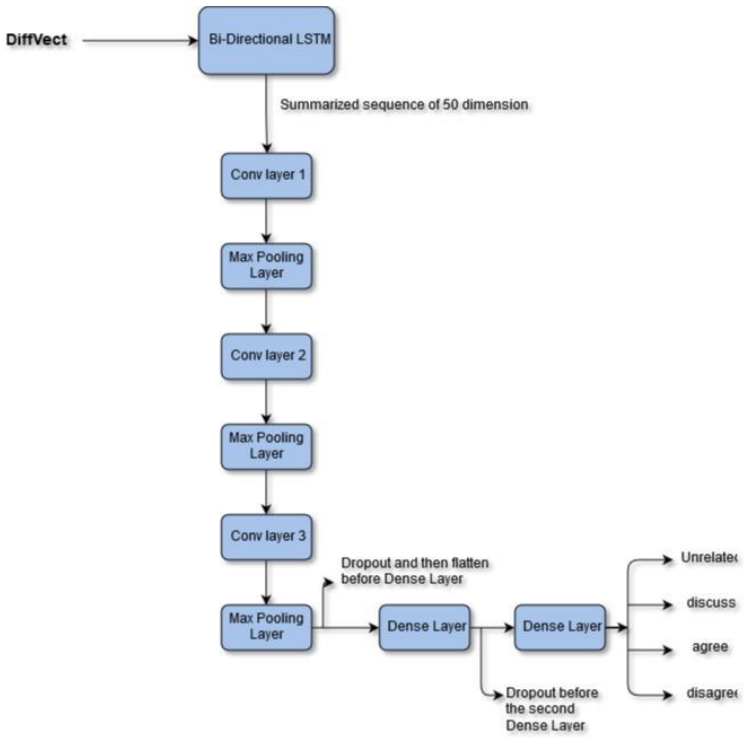
For the activation units in the CNN, we use ReLU in our model since as shown in many papers, ReLU is useful as it can speed up the training and sometimes produces more accurate results.

6. Parameters needed to be optimized

Apart from these, there are certain parameters that are to be optimized:

1. Number of Bi-directional LSTM units
2. Dimension of Kernels in each of the Convolutional layer
3. Batch Size for training the model

The following flowchart display the overall structure of our model:



D. Implementation

We did a grid search on the parameters to get the best possible models. Parameters for grid search were:

1. Number of Bi-LSTM: [32,64]
2. Dimension of kernel: [3,5,7]
3. Batch sizes: 256, 128, 64

The training set was split into training and validation with a 20% split.

We had the following callbacks for the model:

1. Early stop: Stop training if the validation loss doesn’ t decrease for 10 consecutive iterations
2. Checkpoint: Save the weights every time we get better results.

Some of the Adam’ s default parameters:

Adam uses a default learning rate of 0.001, beta_1(first order momentum) = 0.9 and beta_2 (second order momentum) = 0.999 as a starting point and then dynamically adjusts it. Also, I left the decay parameters as default = 0, because initially the model will learn the majority class and we don' t want a decayed learning when it is learning to classify other classes.

Our grid search optimization consisted of two steps

1. Find the best model among the following with a batch size of 100

a. BI-LSTM units = 32, kernel size = 3
b. BI-LSTM units = 32, kernel size = 5
c. BI-LSTM units = 32, kernel size = 7
d. BI-LSTM units = 64, kernel size = 3
e. BI-LSTM units = 64, kernel size = 5
f. BI-LSTM units = 64, kernel size = 7

2. For the best model from above, find the best batch size from 64,128 and 256

E. Model Evaluation and Results

1. Model Evaluation:

We used the following metrics to evaluate our model

1. Mean accuracy
2. Mean Fscore
3. Inverse Weighted F-score: Sum of f-score of each class weighted by inverse of its sample representation, so majority class will have the lowest weight.
4. Mean Precision
5. Mean Recall

Since nearly 73% of the data is the unrelated (the majority class), even if we predict everything as unrelated we still can get 73% accuracy. Because of this imbalances in the data, we will focus more on the mean fscore, inverse weighted fscore, precision and recall rather than only focusing on Accuracy.

2. Results:

The training set was split into training and validation set. Grid search was performed and following were found to be the best optimized parameters:

1. Number of BI-LSTM units = 32
2. Size of kernel in CNN = 5
3. Batch size = 128

Models were trained for 100 epochs.

Following are the results on the Validation/ Test data

Model Architecture	Mean accuracy	Mean Precision	Mean Recall	Mean F-score	Inv - Wt F-Score
BI-Lstm:32, Kernels:3	0.8616	0.6878	0.5206	0.5505	0.3364
BI-Lstm:32, Kernels:5	0.8788	0.7110	0.5857	0.6149	0.4552
BI-Lstm:32, Kernels:7	0.8760	0.7750	0.5458	0.5616	0.2825
BI-Lstm:64, Kernels:3	0.8739	0.6988	0.5786	0.6076	0.4570
BI-Lstm:64, Kernels:5	0.8754	0.7461	0.5676	0.5909	0.3777
BI-Lstm:64, Kernels:7	0.8740	0.6652	0.5741	0.5899	0.3718

Table showing the grid search scores for different models. In this grid search we are optimizing for the Bi-Lstm units and the kernel dimension. The batch size is kept constant at 100. Acronyms are: Inv - Wt means Weighted fscore where the weights are inversely proportional to the sample representation

We will select the model which has the maximum Mean fscore and the Maximum Inverse weighted fscore. The reason being that the accuracy can be swayed by the majority class. It is evident that the two models: (BI-Lstm:32, Kernels:5) and (BI-Lstm:64, Kernels:3) performs the best and have scores nearly equal. So, we

will select the model (BI-Lstm:32, Kernels:5) as the best model as it is less complex that the other one and hence will suffer less from error due to variance.

To find the batch size, we ran another grid search on the model (BI-Lstm:32, Kernels:5) with different batch sizes.

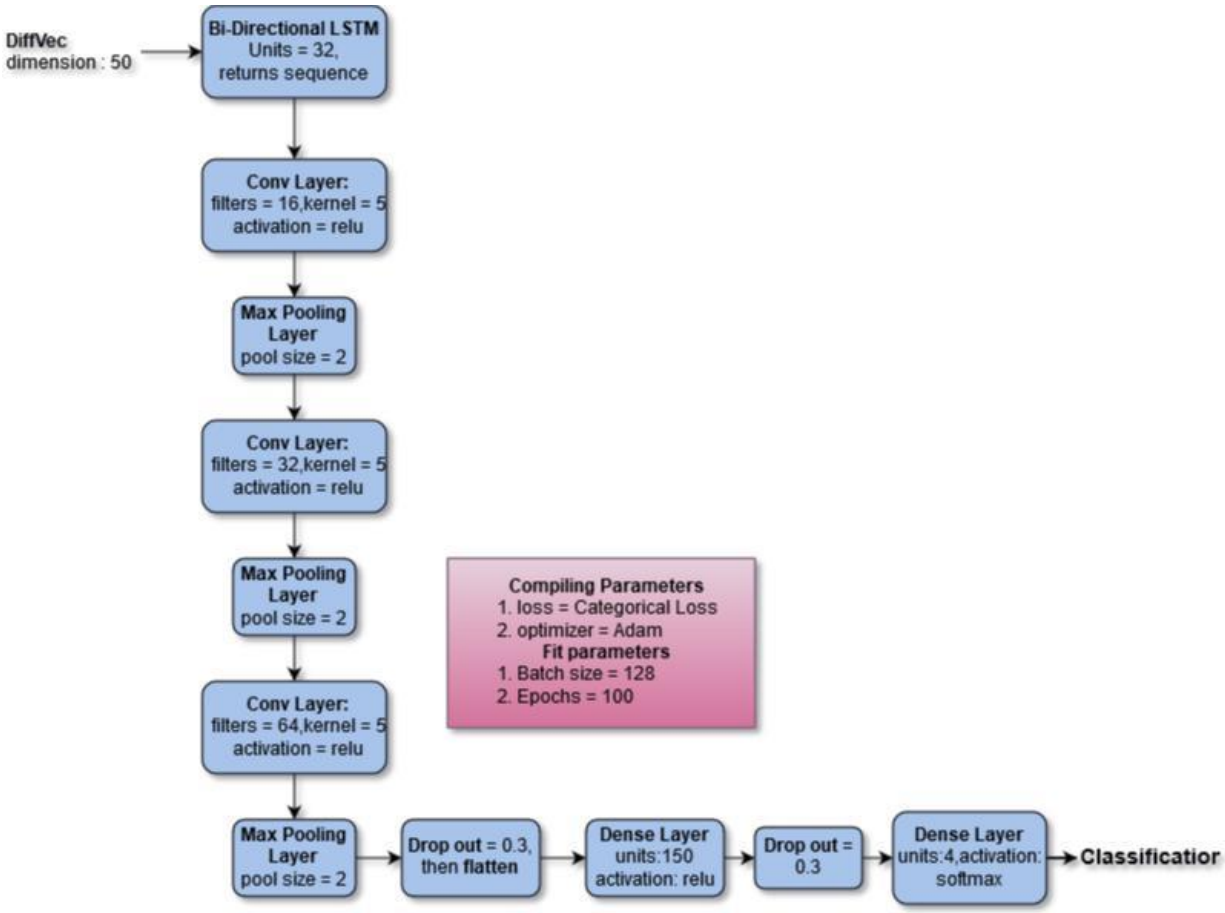
Following are the results on the Validation/ Test data

Model Architecture	Mean accuracy	Mean Precision	Mean Recall	Mean F-score	Inv - Wt F-Score
Batch size: 64	0.8727	0.6952	0.5756	0.6026	0.4306
Batch size: 128	0.8788	0.7110	0.5857	0.6149	0.4552
Batch size: 256	0.8640	0.6402	0.5373	0.5415	0.2141

Scores for different batch sizes for the model with 32 Bidirectional Lstm units and kernel dimension as 5

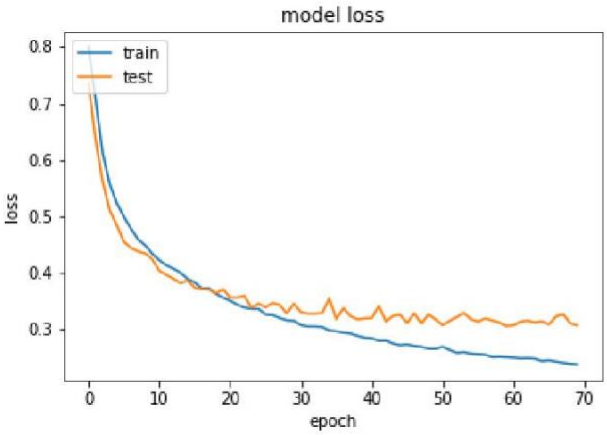
So, after the exhaustive grid search, the final optimized parameters are

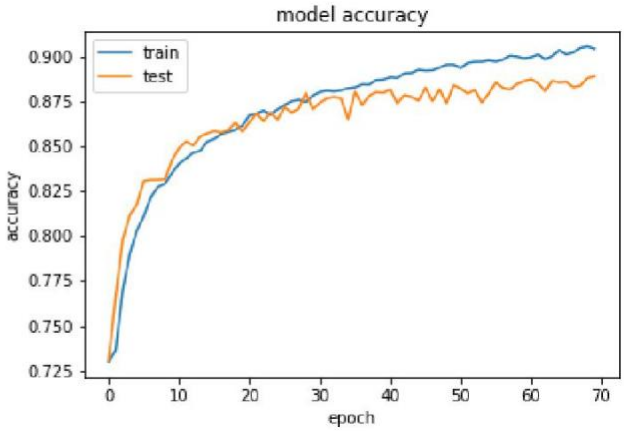
1. **Bidirectional LSTM units: 32**
2. **Dimension of Kernel: 5**
3. **Batch size for training: 128**



The Final architecture with optimized parameters

Following are the Model loss and Model accuracy for the Training and Validation test cases with respect to epochs

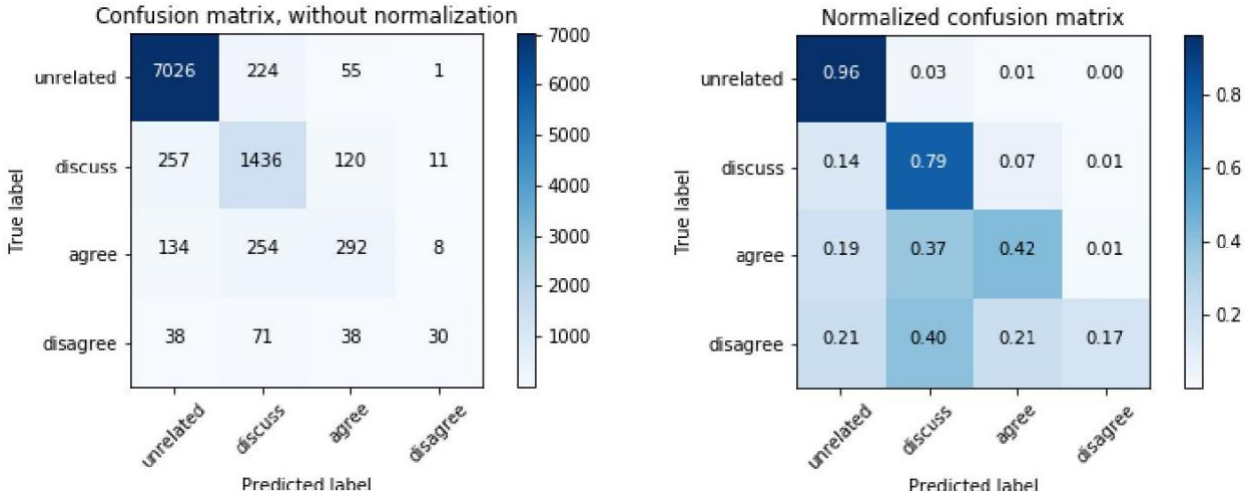




As we can see, that the model starts to converge roughly after 30 to 40 epochs. The dropouts and the regularization prevented the overfitting and didn' t let the accuracy decrease. So, we can be sure that this is the best model given the sample representation as diffVec.

F. Conclusion

Free form visualization



One thing to note here is that if something is classified as fake/disagree, we should have high confidence for it. We just can' t say that an article-headline is fake with some probability. This means that we should have a high precision in classifying something as disagree. If this is not the case, and an article-headline pair that is not fake and the model classifies it as Fake, then the model would be criticized. We can accept classifying some pair as not Fake when in case they are but cannot accept the opposite at any cost.

From the confusion plot, we can see that 20 out of 9818 pair were classified as disagree when actually they were not. This is 0.2%, which means we have 98.0% confidence in classifying something as disagree/fake.

We also got a good mean f-score of 0.6149 which means that the model is not just predicting everything as unrelated (majority class).

G. A Useful Detector

We would like to build up a practically useful detector to identify the relationship between the headlines and body articles based on the optimized model we built before.

The input: two csv.file

a. headline.csv: a file including headline
b. body.csv: a file including body article corresponding to the headline in headline.csv

The output: The classification of relationship between the headline and its corresponding body article.

Implementation:

1. Read the headline and body article from the csv file.

Here is an example to show how the headlines and body articles are scored in Python.

Headline (An example with 6 rows of headline in headline.csv):

[['police find mass graves with at least 15 bodies near mexico town where 43 students disappeared after police clas', "Polic e find mass graves with at least '15 bodies' near Mexico town where 43 students disappeared after police clash"], ['hundreds of palestinians flee floods in gaza as israel opens dam', 'Hundreds of Palestinians flee floods in Gaza as Israel opens dams'], ['christian bale passes on role of steve jobs actor reportedly felt he wasn t right for par', "Christian Bale passes on role of Steve Jobs, actor reportedly felt he wasn't right for part"], ['hbo and apple in talks for 15 month apple tv streaming service launching in apri', 'HBO and Apple inTalks for $15/Month Apple TV Streaming Service Launching in April'], ['spider burrowed through tourist s stomach and up into his ches', "Spider burrowed through tourist's stomach and up into his chest"]]

Body Article (An example with 6 rows of body article in body.csv. For space here, we only show one of them):

\nGovernment spokeswoman Rosario Murillo said a committee formed by the government to study the event determined it was a "relatively small" meteorite that "appears to have come off an asteroid that was passing close to Earth." House-sized asteroid 2014 RC, which measured 60 feet in diameter, skimmed the Earth this weekend, ABC News reports.

2. Convert the words into word vectors. Like the algorithm shown before, the output will be diffVec (bodyVec – titleVec)

3. Use the optimized model to do the prediction.

Code:

```
def create_model(num_classes,num_lstm_units,X_train_shape,kernel_size):
    # create model
    model = Sequential()
    model.add(Bidirectional(LSTM(units = num_lstm_units ,return_sequences=True),
                        input_shape=(X_train_shape[1],X_train_shape[2])))

    model.add(Conv1D(filters=16, kernel_size=kernel_size, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))

    model.add(Conv1D(filters=32, kernel_size=kernel_size, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))

    model.add(Conv1D(filters=64, kernel_size=kernel_size, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))

    model.add(Dropout(0.3))
    model.add(Flatten())

    model.add(Dense(150, activation='relu'))
    model.add(Dropout(0.4))

    model.add(Dense(num_classes,activation = 'softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['categorical_accuracy'])
    model.summary()
    return model
```

```
lstm=32
ker=5
batch=128
num_classes = dummy_y.shape[1]
input_shape = X_train.shape
class_weight = compute_class_weight('balanced', np.unique(y), y)
class_weight_norm = class_weight/np.linalg.norm(class_weight)
model = create_model(num_classes =num_classes,num_lstm_units = lstm,X_train_shape = input_shape,kernel_size = ker)
model.load_weights('results/batch/32_5_128.weights.best.hdf5')
y_pred = model.predict_classes(x_pred)

print(y_pred)
```

The prediction result: [1 0 0 1 0 0]

Here, we define:

0→unrelated; 1→discuss; 2→agree; 3→disagree

H. Reference

Data source: https://fakenewschallenge.org

Word Vector: Global Vectors for Word Representation

from https://nlp.stanford.edu/projects/glove/

[1] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Advances in neural information processing systems, pages 3104–3112, 2014.

[2] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. Citeseer.

[3] X. Sun et al. (Eds.): Comparative Study of CNN and RNN for Deep Learning Based Intrusion Detection System ICCCS 2018, LNCS 11067, pp. 159–170, 2018.