Assignment 3

Sadid Rafsun Tulon
300209875

1 Introduction

Paraphrase detection is an important part of intent detection which can help the assistant and chatbots. But para phrase detection is a very difficult task as there might be two paraphrase sentences which have almost no common words like “He has tons of stuff to throw away.” and “He needs to get rid of a lot of garbage. One the other hand, paraphrase identification is that not all words are always matched, even if they are not paraphrases. The sentences "I bought mango for lunch" and " I brought mango for lunch," for example, have only one letter difference but they are not paraphrase. As a result, finding paraphrases is a challenging process, as the key goal is to comprehend the context and meanings, not merely match words.

2 Dataset

I utilised the train.data, dev.data, and test.data files in this assignment for both training and testing the data from SemEval-2015 Task 1: Paraphrase and Semantic Similarity on Twitter (PIT)[1]. I utilised the entire file, which had 13063, 4727, and 972 sentences, respectively. The dataset contained graded assessments in the form of a tuple, each of which was based on five votes. I have used a simple python script to preprocess the data files. In train.data and dev.data "(3, 2)", "(4, 1)" and "(5, 0)" are turned into 1 and, "(1, 4)" and "(0, 5)" are turned into 0. Rest of the data are discarded. On test.data “4” and “5” are turned into 1 and, “0”,”1” and “2” are turned into 0 while rest of the data are discarded.

Examples of paraphrases and non-paraphrases are given in Table 1 and Table 2, respectively.

Sentence 1	Sentence 2
THANK YOU SO MUCH RAFA BENITEZ	Credit where credits due to Rafa Benitez
Chara is a disgrace to the NHL	Chara is just a big goon
Russell Westbrook is gonna have surgery on his knee	they just said Russell westbrook gonna have surgery
There s always a scar	Behind every scar is an untold story
The steelers pick Shamarko Thomas screams Pittsburg	Steelers took S Shamarko Thomas with pick from Syracuse

Table 1: Paraphrases

Sentence 1	Sentence 2
I mean look at shaq	How the hell does Shaq fit in that Buick
Scrappy And Shay Look Like Siblings	Momma Dee and Shay are super ratchet
FOLLOW ME PLEASE GO ON SIMON	Simon please could you tweet PrayForKrista
Ive got to get starbucks first	45 to Starbucks in my mailbox
I hate the Rangers so much	I am watching the NY Rangers game

Table 2: Non-paraphrases

3 Methods

To implement paraphrase detection, I have used three algorithms, first one being cosine similarity then neural network binary classifier. Last algorithm is slightly modified version of second algorithm to improve performance. For calculating precision and recall, I have written a simple method. Formula for precision is true positive / (true positive + false positive) and formula for recall is true positive / (true positive + false negative).

3.1 Algorithm 1

For first algorithm, I have used Cosine Similarity method[3]. First, I have converted the sentences to vector using Bag of Words method. I have used CountVectorizer from sklearn library for this[2]. This gives me two arrays for two sentences. For example, Bag of Words for "I am driving car" and "I am driving truck" would be ["I", "am", "driving", "car", "truck"]. Then the two sentences would be [1,1,1,1,0] and [1,1,1,0,1] respectively. Then I have used cosine similarity on them. The formula of cosine similarity is

$$\frac{\sum x * y}{\sqrt{(\sum x * x)(\sum y * y)}}$$

Here x and y are the two-matrix derived from the sentences. Here the result is .75. From my experiment I have observed that setting threshold value to .35 gives the best result. I have made a script to read sentences as x1_dev and x2_dev and, label as y_dev from preprocessed dev.data file. Then I made a loop to match the sentences and store the results in a variable called y_pred. After that I calculated recall and precision from comparing y_dev (true result) and y_pred (predicted result)

3.2 Algorithm 2

The second algorithm is Neural Network method. Neural Network are designed to recognize patterns like human brains. Neural networks are being used for speech and image recognition, spam email filtering [5]. As paraphrase detection is somewhat similar to text classification and Neural Network is very good at it, Neural Network should be ideal choice for paraphrase detection.

Like first algorithm the sentences are turned into Bag of Words. But this time I used preprocessed train.data to form matrix before supplying them to Neural Network function. I have used 1 dense layer with 200 units and ReLu for input, 6 dense layer with 200 units each and ReLu for the hidden layers and, 1 unit and sigmoid for output layer for the Neural Network. I have also used 10 epochs for training the data. Batch size for training is 16. From my observation, this configuration gave me the best result.

After training the model, I used preprocessed dev.data to form Bag of Words matrix then used the model to predict results and got an array which I stored in y_pred. Lastly I calculated recall and precission using y_pred and y_dev from preprocessed dev.data.

Unfortunately, this algorithm gave worst result out of all three algorithms.

3.3 Algorithm 3

The third algorithm is slightly modified version of previous algorithm. Results of previous algorithm was not great. So I tried to use GloVe Embedding to get better result.

GloVe is mainly focused on deriving the relationship between the words from statistics. Unlike Bag of words technique, GloVe tells how often a particular word pair occurs together [6]. This gives more meaning to the matrix as well as better weight for training the Neural Net.

So in this algorithm, instead of using Bag of Words method, I used GloVe to make matrixes before supplying them to the Neural Network function. I have used "glove.6B.200d" dataset. If we take a look into the dataset we can see something like "after 0.38315 -0.3561 -0.1283 -0.19527 0.047629... ". This dataset contains each word and its representation vector. I replaced the words in the train.data and dev.data to vector representation before turning them into matrixes. Any word which is not present in "glove.6B.200d" is given value of 0.

I used exact same configuration as Algorithm 2 for the Neural Network model. Then I supplied the matrix pair to in the model and trained the model. After training the model, I supplied inputs from dev.data to make prediction. The model then gave a prediction array which I stored as y_pred.

4 Results

For evaluation two metrics have been evaluated, namely precision and recall using my own method. The results of each algorithm are shown in Table 3.
The best performance is given by the first algorithm which is based on Cosine Similarity.

Algorithm	Precision	Recall
1	0.521	0.462
2	0.268	0.413
3	0.462	0.418

Table 3: Precision, Recall and F1-score of all three algorithms

I further tested the algorithm on test set. Here, I created a loop to iterate over preprocessed test.data and calculated the possible results for all pairs of sentences and matched those results with true result. The test results were:

Precision	Recall
0.345	0.289

Table 4: Results in test data for the best algorithm

From the results, it is quite surprising that the Cosine Similarity gave the best result. Neural Network performed worst in my experience despite my best effort. GloVe embedding gave slightly better scores.

But Cosine Similarity is not very good paraphase detector. It will give positive result to sentences with similar word such as “I ate candy” and “I hate candy” , however these sentences are not paraphrases. While paraphases like “I am hungry” and “Need to eat food” will get negative score.

5 Conclusions and Future work

From the experiments, Cosine Similarity gave the best result. But this method is not that much reliable as it relies heavily on match the strings. This is not ideal for paraphrase detection as only sentences with similar word will be identified as paraphrase. Neural Network should have more potential but did not give result up to mark. Even with GloVe embeddings the result was underwhelming at best. More experiment should be done on this method. In SemEval (2015) a paraphrase detector algorithm based on SVM classifier with some data preprocessing secured first rank so using this method should yield to better result[4].

6 References

[1] COCOXU, SemEval-2015 Task 1: Paraphrase and Semantic Similarity in Twitter (PIT), Edited Nov, 2020, GitHub repository, https://github.com/cocoxu/SemEval-PIT2015

[2] Janakiev N. Oct, 2018., URL https://realpython.com/python-keras-text-classification/

[3] Wikipedia. URL https://en.wikipedia.org/wiki/Cosine_similarity

[4] Eyecioglu, A. and Keller, B. (2015). ASOBEK: Twitter paraphrase identification with simple overlap features and SVMs. In Proceedings of SemEval.

[5] DeepAi URL https://deepai.org/machine-learning-glossary-and-terms/neural-network

[6] Wikipedia. URL https://en.wikipedia.org/wiki/GloVe_(machine_learning)