Minor Project Report on

Paraphrase Detection using Deep Learning

A Aditya (15IT201)

Under the Guidance of,

Dr. Sowmya Kamath S

Department of Information Technology, NITK Surathkal

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

May 2018

Certificate

This is to certify that the Minor Project entitled "Paraphrase Detection using Deep Learning" is a bonafide work carried out, under my guidance by A Aditya student of VI Sem B.Tech (IT) at the Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the academic semester of Jan - May 2018 in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, at NITK Surathkal.

Place:

Date: *(Name and Signature of Project Guide)*

Abstract

Paraphrase detection is the task of examining two sentences and determining whether they have the same meaning. Previous approaches either match sentences from a single direction or only apply single granular (word by word or sentence by sentence) matching. In this work, two types of bilateral multi-perspective matching (BiMPM) models are used. One with is the model as given in and the other is an attention based variant of this model. Given two sentences P and Q, both the models first encode them with a BiLSTM encoder. Next, the encoded sentences are matched sentences in two directions P against Q and Q against P . In each matching direction, each time step of one sentence is matched against all time steps of the other sentence from multiple perspectives. Then, another BiLSTM layer is utilized to aggregate the matching results into a fixed-length matching vector. This is followed by an attention layer in case of the attention model. Finally, a decision is made through a fully connected layer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Research on paraphrasing methods typically aims at solving three related problems: (1) recognition (i.e. to identify if two textual units are paraphrases of each other), (2) extraction (i.e. to extract paraphrase instances from a thesaurus or a corpus), and (3) generation (i.e. to generate a reference paraphrase given a source text). In this paper, we focus on the paraphrase detection problem. Paraphrase detection determines whether two phrases of arbitrary length and form capture the same meaning. If the two sentences convey the same meaning it is labelled as paraphrase, otherwise non-paraphrase. Identifying paraphrases is an important task that is used in information retrieval, question answering, text summarization, plagiarism detection and evaluation of machine translation , among others. We focus on applying the task of paraphrase detection to the medical domain where it has not applied before.

1.1 Challenges

The main challenge in paraphrase detection is that the identification of a paraphrase relationship between two sentences requires an analysis at multiple levels of granularity. Most work on paraphrase identification has focused on only one level of granularity: either on low-level features or on the sentence level. Also, paraphrase detection has not been applied to the medical domain yet. Medical sentences are very different from normal sentences because of the use of high-level medical terms.

1.2 Motivation

Identifying paraphrases is an important task that is used in information retrieval, question answering, text summarization, plagiarism detection and evaluation of machine translation, among others. Question paraphrase identification is also a widely useful NLP application. For example, in question and answer (QA) forums on the Web, there are vast numbers of duplicate questions. Identifying these duplicates and consolidating their answers increases the efficiency of such QA forums. Moreover, identifying questions with the same semantic content could help Web-scale question answering systems that are increasingly concentrating on retrieving focused answers to users queries. Another domain where paraphrase detection could be applied is in the field of medicine. For example, different doctors' might write different kinds of prescriptions for the same disease. Identifying paraphrases in prescriptions could help the patient who might have difficulty in understanding a prescription and would also help in reducing redundancies. In this project we focus on applying the task of paraphrase detection to the medical domain.

Chapter 2

Literature Review

2.1 Previous Work

With the renaissance of neural network models two types of deep learning frameworks were proposed for the task of sentence matching which formed the basis for future work in paraphrase detection. The first framework is based on the "Siamese" architecture [3]. In this framework, the same neural network encoder (e.g., RNN) is applied to two input sentences individually, so that both of the two sentences are encoded into sentence vectors in the same embedding space. Then, a matching decision is made solely based on the two sentence vectors [2]. The advantage of this framework is that sharing parameters makes the model smaller and easier to train, and the sentence vectors can be used for visualization, sentence clustering and many other purposes [14]. However, a disadvantage is that there is no explicit interaction between the two sentences during the encoding procedure, which may lose some important information.

To deal with this problem, a second framework matching-aggregation has been proposed [12]. Under this framework, smaller units (such as words) of the two sentences are firstly matched, and then the matching results are aggregated (by a CNN or LSTM) into a vector to make the final decision. The new framework captures more interactive features between the two sentences, therefore it acquires significant improvements. However, the previous "matching aggregation" approaches still have some limitations. First, some of the approaches only explored the word-by-word matching [8], but ignored other granular matchings (e.g., phrases); Second, the matching is only performed in a single direction (e.g., matching P against Q) [11], but neglected the reverse direction.

Socher et al [10] proposed a method based on dynamic pooling and unfolding recursive autoencoders (RAE). RAE phrase features compares both single word vectors as well as phrases and complete syntactic trees. But relying on parsing has limitations for noisy text and for other applications in which highly accurate parsers are not available.

In the medical field as well the task paraphrase detection and generation has been explored. Grabar et al [5] proposed an automatic method based on the morphological analysis of terms and on text mining for finding the paraphrases of technical terms in medicine. Depending on the semantics of the terms, error rate of the extractions ranged between 0 and 59. But one of the biggest demerits of this work is the lack of constraints on the extracted segments.

2.2 Outcome of Literature Review

Paraphrase detection requires comparison of sentences at multiple levels of granularity. Thus, in addition to words phrases also need to be matched and compared for better results. Also matching the two sentences in both directions instead of just one using a bidirectional LSTM could help in learning better representations from the model.

2.3 Problem Statement

Given 2 sentences identifying if they are paraphrases or not, in other words examining two sentences and determining whether they have the same meaning using a deep learning model.

2.4 Research Objectives

1. To implement existing models for paraphrase detection.
2. To improve existing models for paraphrase detection.
3. To develop a hybrid model for clinical paraphrase detection with clinical data.

Chapter 3

Methodology and Framework

3.1 System Architecture

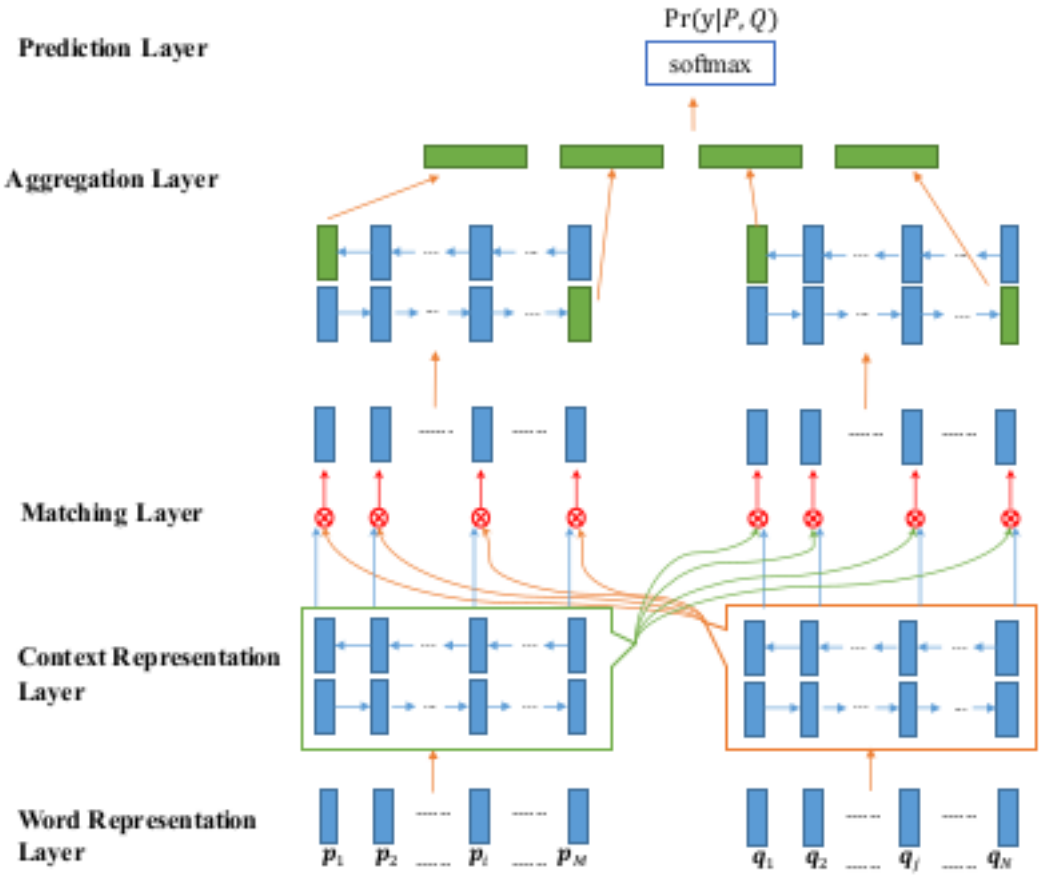

Figure 3.1: Architecture of the Base Model [13]

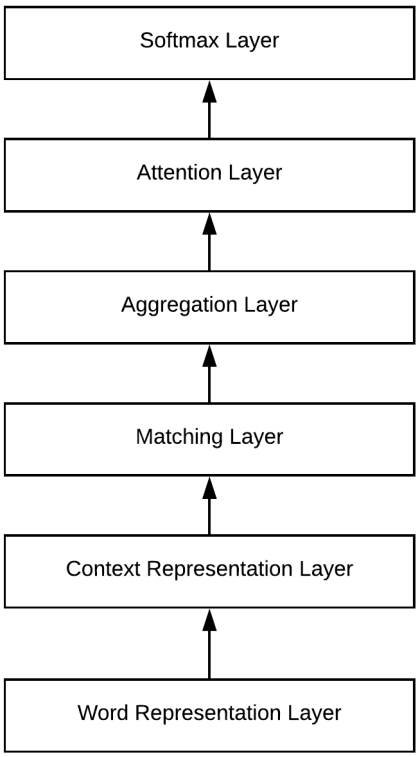

Figure 3.2: Architecture of the Attention based model

3.2 Base Model Overview

This section and the next section give an overview of the base model as given in [13]

3.2.1 Word Representation Layer

The goal of this layer is to represent each word in P and Q with a d-dimensional vector. We construct the d-dimensional vector with pretrained glove embeddings. The output of this layer are two sequences of word vectors P : $p_1, ..., p_M$ and Q : $q_1, ..., q_N$.

3.2.2 Context Representation Layer

The purpose of this layer is to incorporate contextual information into the representation of each time step of P and Q. We utilize a bi-directional LSTM (BiLSTM) to encode contextual embeddings for each time-step of P.

$$\overrightarrow{h_i}^p = \overrightarrow{LSTM}(\overrightarrow{h_{i-1}}^p, p_i) \quad i = 1, 2, ..., M$$

$$\overleftarrow{h_i}^p = \overleftarrow{LSTM}(\overleftarrow{h_{i+1}}^p, p_i) \quad i = M, M-1, ..., 1$$

We apply the same BiLSTM to encode Q :

$$\overrightarrow{h_j}^q = \overrightarrow{LSTM}(\overrightarrow{h_{j-1}}^q, q_j) \quad j = 1, 2, ..., N$$

$$\overleftarrow{h_j}^q = \overleftarrow{LSTM}(\overleftarrow{h_{j+1}}^q, q_j) \quad j = N, N-1, ..., 1$$

3.2.3 Matching Layer

The goal of this layer is to compare each contextual embedding (time-step) of one sentence against all contextual embeddings (time-steps) of the other sentence.The output of this layer are two sequences of matching vectors where each matching vector corresponds to the matching result of one time-step against all time-steps of the other sentence. More details about the matching layer is given in section 3.3 .

3.2.4 Aggregation Layer

This layer is employed to aggregate the two sequences of matching vectors into a fixed-length matching vector. We utilize another BiLSTM model, and apply it to the two sequences of matching vectors individually. Then, we construct the fixed-length matching vector by concatenating vectors from the last time-step of the BiLSTM models

3.2.5 Prediction Layer

The purpose of this layer is to evaluate the probability distribution Pr(y|P, Q). To this end, we employ a two layer feed-forward neural network to consume the fixed-length matching vector, and apply the softmax function in the output layer.

3.3 Multi Perspective Matching

First, a multi-perspective cosine matching function f_m is defined to compare two vectors:

$$m = f_m(v_1, v_2; W)$$

where v_1 and v_2 are two d-dimensional vectors, W is a trainable parameter with the shape l * d, l is the number of perspectives, and the returned value m is a l-dimensional vector m = $[m_1, ..., m_k, ..., m_l\ ]$. Each element $m_k\ \epsilon$ m is a matching value from the k-th perspective, and it is calculated by the cosine similarity between two weighted vectors

$$m_k = cosine(W_k \circ v_1, W_k \circ v_2)$$

where $\circ$ is the element-wise multiplication, and W_k is the k-th row of W , which controls the k-th perspective.

Second, four matching functions are defined which are given in figure 3.2 :

1. **Full Matching** : In this strategy, each forward (or backward) contextual embedding $\overrightarrow{h_i}^p$ (or $\overleftarrow{h_i}^p$) is compared with the last time step of the forward (or backward) representation of the other sentence $\overrightarrow{h_N}^q$ (or $\overleftarrow{h_1}^q$).

$$\overrightarrow{m_i}^{full} = f_m(\overrightarrow{h_i}^p, \overrightarrow{h_N}^q; W^1)$$

$$\overleftarrow{m_i}^{full} = f_m(\overleftarrow{h_i}^p, \overleftarrow{h_1}^q; W^2)$$

2. **Max Pooling Matching** : In this strategy, each forward (or backward) contextual embedding $\overrightarrow{h_i}^p$ (or $\overleftarrow{h_i}^p$) is compared with every forward (or backward) contextual embeddings of the other sentence $\overrightarrow{h_i}^q$ (or $\overleftarrow{h_i}^q$) for j ϵ (1...N), and only the maximum value of each dimension is retained.

$$\overrightarrow{m_i}^{max} = \max_{j\epsilon(1..N)} f_m(\overrightarrow{h_i}^p, \overrightarrow{h_j}^q; W^3)$$

$$\overleftarrow{m_i}^{max} = \max_{j\epsilon(1..N)} f_m(\overleftarrow{h_i}^p, \overleftarrow{h_j}^q; W^4)$$

3. **Attentive Matching** : First calculate the cosine similarities between each forward (or backward) contextual embedding $\overrightarrow{h_i}^p$ (or $\overleftarrow{h_i}^p$) and every forward (or backward) contextual embeddings of the other sentence $\overrightarrow{h_j}^q$ (or $\overleftarrow{h_j}^q$) :

$$\overrightarrow{a_{i,j}} = cosine(\overrightarrow{h_i}^p, \overrightarrow{h_j}^q) \quad j = 1, 2, ..., N$$

$$\overleftarrow{a_{i,j}} = cosine(\overleftarrow{h_i}^p, \overleftarrow{h_j}^q) \quad j = 1, 2, ..., N$$

Then, take $\overrightarrow{a_{i,j}}$ (or $\overleftarrow{a_{i,j}}$) as the weight of $\overrightarrow{h_j}^q$ (or $\overleftarrow{h_j}^q$) and calculate an attentive vector for the entire sentence Q by weighted summing all the contextual embeddings of Q:

$$\overrightarrow{h_i}^{mean} = \frac{\sum_{j=1}^{N} \overrightarrow{a_{i,j}}.\overrightarrow{h_j}^q}{\sum_{j=1}^{N} \overrightarrow{a_{i,j}}}$$

$$\overleftarrow{h_i}^{mean} = \frac{\sum_{j=1}^{N} \overleftarrow{a_{i,j}}.\overleftarrow{h_j}^q}{\sum_{j=1}^{N} \overleftarrow{a_{i,j}}}$$

Finally, match each forward (or backward) contextual embedding of $\overrightarrow{h_i}^p$ (or $\overleftarrow{h_i}^p$) with its corresponding attentive vector:

$$\overrightarrow{m_i}^{att} = f_m(\overrightarrow{h_i}^p, \overrightarrow{h_i}^{mean}; W^5)$$

$$\overleftarrow{m_i}^{att} = f_m(\overleftarrow{h_i}^p, \overleftarrow{h_i}^{mean}; W^6)$$

4. **Max Attentive Matching** : This strategy is similar to the Attentive-Matching strategy. However, instead of taking the weighed sum of all the contextual embeddings as the attentive vector, pick the contextual embedding with the highest cosine similarity as the attentive vector. Then, match each contextual embedding of the sentence P with its new attentive vector.

3.4 Attention Model

Attention is a mechanism that was developed to improve the performance of the Encoder-Decoder RNN on machine translation. A potential issue with encoder-decoder approaches in machine translation is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector. This may make it difficult for the neural network to cope with long sentences. With an attention mechanism

we no longer try encode the full source sentence into a fixed-length vector. Rather, we allow the decoder to "attend" to different parts of the source sentence at each step of the output generation. Importantly, we let the model learn what to attend to based on the input sentence and what it has produced so far. Attention was proposed as a method to both align and translate.

Here the attention mechanism proposed in [16] is used. The main idea is that not all words contribute equally to the representation of the sentence meaning. Hence, we introduce attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. The equations used are mentioned below:

$$u_{it} = tanh(W_w h_{it} + b_w)$$

$$\alpha_{it} = \frac{exp(u_{it}^\intercal u_w)}{\sum_t exp(u_{it}^\intercal u_w)}$$

$$s_i = \sum_t \alpha_{it} h_{it}$$

The Attention based model architecture is given in figure 3.2 . The sequence vectors obtained after the aggregation layer (h_i) are passed through a one-layer MLP to get u_{it} as a hidden representation of h_{it}, then we measure the importance of the word as the similarity of u_{it} with a word level context vector u_w and get a normalized importance weight α_{it} through a softmax function. After that, we compute the sentence vector s_i which replaces h_i, as a weighted sum of the word annotations based on the weights.

Chapter 4

Work done

4.1 Experimental Framework

The experiments were conducted on a system with 56 Intel Xeon CPU E52680 cores and Nvidia Tesla M40 GPU with 3072 Nvidia CUDA cores and 24 GB of GDDR5 video memory. We set the hidden size as 100 for all BiLSTM layers. We apply dropout to every layer, and set the dropout ratio as 0.1. To train the model, we minimize the cross entropy of the training set, and use the ADAM optimizer to update parameters. We set the learning rate as 0.0001. During training, we do not update the pretrained word embeddings. The number of perspectives was kept at 20.

4.2 Implementation

Our implementation is divided into 4 tasks. TensorFlow Deep learning framework [1] was used for implementation. The first task is the implementation of the base model as given in [13] and comparing the effect of LSTM and GRU memory units in the model. The dataset used for this task is the Microsoft Research paraphrase corpus (MSRP) introduced by Dolan et al. [4]. The dataset consists of 5,801 sentence pairs. The average sentence length is 21, the shortest sentence has 7 words and the longest 36. 3,900 are labeled as being in the paraphrase relationship. We use the standard split of 4,076 training pairs (67.5 of which are paraphrases) and 1,725 test pairs (66.5 paraphrases). We found comparable performance of both models. A very slight improvement in accuracy for the GRU model when compared to the LSTM model. This could be attributed to the GRU

Table 4.1: Performance comparison of LSTM and GRU

Model	**Train Accuracy(%)**	**Test Accuracy(%)**
LSTM	73.22	73.37
GRU	70.22	73.5

model needing lesser parameters to train and hence doing better on a smaller dataset.

The second task was to compare the effect of word embeddings on the performance of the models. So, we compared the pretrained glove embeddings LSTM model with the pretrained word2vec LSTM model on the Microsoft Research paraphrase corpus dataset. We found that the glove model outperformed the word2vec model by a big margin.

The third task is the implementation of the attention based model and comparing it with the base model. Here, the attention mechanism proposed in [16] is used. The attention model is implemented using both LSTM and GRU memory units using glove pretrained embeddings on the Microsoft Research paraphrase corpus dataset. The attention model performed better than the base model for both LSTM and GRU cells.

The final task was to apply paraphrase detection to the medical domain. For this, we developed a paraphrase corpus manually from the clinical notes of the i2b2 dataset. This medical paraphrase dataset consists of 150 pairs of sentences in the training set and 60 pairs of sentences in the test set. We compared the LSTM and GRU variants off both the base model and the attention model on this dataset. We found that the base model performed better here contrary to the MSRP dataset.

4.3 Results and Analysis

The results of all the 4 tasks explained in the Implementation section are compiled here. Table 4.3 contains results from previous work.

The first task is the implementation of the base model as given in [13] and comparing the effect of LSTM and GRU memory units in the model. We found comparable performance of both models. A very slight improvement in accuracy for the GRU model when compared to the LSTM model. The results are in table 4.1 .

The second task was to compare the effect of word embeddings on the performance

Table 4.2: Performance comparison of glove and word2vec embeddings

Model	**Train Accuracy(%)**	**Test Accuracy(%)**
Glove LSTM model	73.22	73.37
Word2vec LSTM model	68.57	66.41

Table 4.3: Results from previous work

Model	**Test Accuracy(%)**
Baseline	66.5
Hasan et al. [6]	67.0
Mihalcea et al. [9]	70.3
Socher et al. [10]	76.8
Wang et al. [15]	78.4
Ji et al. [7]	80.4

of the models on the MSRP dataset. We found that the glove model outperformed the word2vec model by a big margin. The results are in table 4.2 .

The third task is the comparison of the attention based model with the base model on the MSRP dataset with glove pretrained embeddings. The attention model performed better than the base model for both LSTM and GRU cells. The results are in table 4.4 .

The final task was to apply paraphrase detection to the medical domain. For this, we developed a paraphrase corpus manually from the clinical notes of the i2b2 dataset. This medical paraphrase dataset consists of 150 pairs of sentences in the training set and 60

Table 4.4: Performance comparison of Base model and Attention model on MSRP dataset

Model	**Train Accuracy(%)**	**Test Accuracy(%)**
Base LSTM model	73.22	73.37
Attention LSTM model	73.42	73.96
Base GRU model	70.22	73.5
Attention GRU model	72.65	73.7

Table 4.5: Performance comparison of Base model and Attention model on Medical dataset

Model	**Train Accuracy(%)**	**Test Accuracy(%)**
Base LSTM model	67.47	71.67
Attention LSTM model	75.2	68.33
Base GRU model	69	70
Attention GRU model	77.13	68.33

pairs of sentences in the test set. We compared the LSTM and GRU variants off both the base model and the attention model on this dataset. We found that the base model performed better here contrary to the MSRP dataset. This could be attributed to the lesser number of sentences to train on for the attention model and hence it could not find appropriate weights to assign to words in a sentence. The results are in table 4.5 .

The Accuracy and loss curves for both base and attention models are given below:

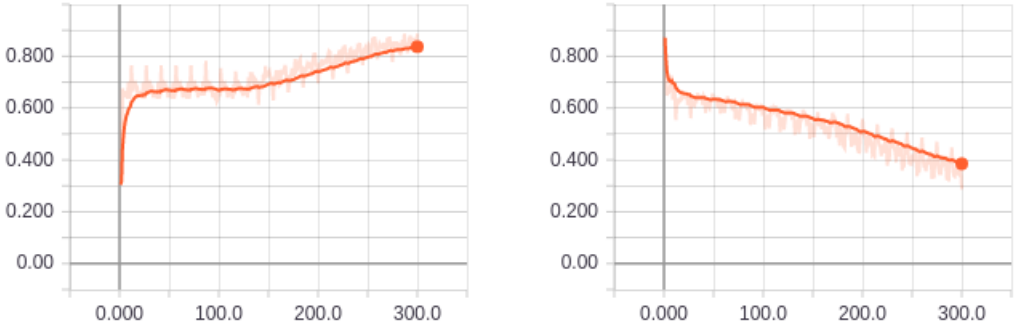

Figure 4.1: Accuracy and loss curves for base model on MSRP

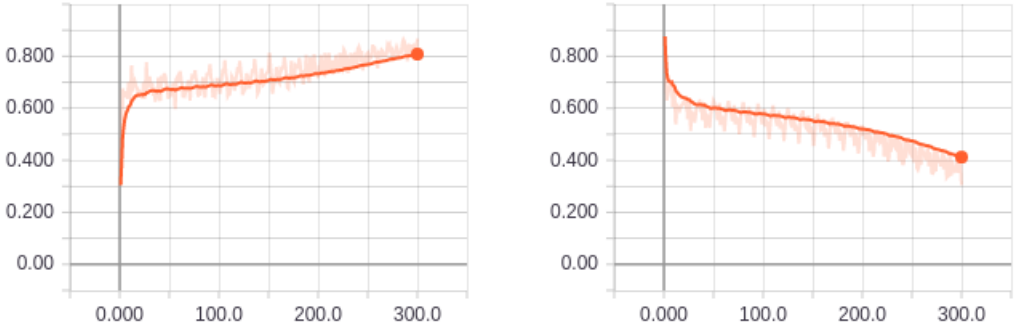

Figure 4.2: Accuracy and loss curves for attention model on MSRP

Chapter 5

Conclusion & Future Work

In this project work the task of paraphrase detection was implemented using 2 deep learning models. The first base model was based on [13] and the second model was a variant of the base model with attention mechanism. For comprehensive evaluation 4 tasks were carried out to determine the best memory cell, best pretrained word embedding, best model in case of MSRP dataset and best model in case of medical dataset. Both lstm and gru cells gave comparable performance. The pretrained glove embeddings performed better than the pretrained word2vec embeddings. The attention based model performed better on the MSRP dataset but the base model performed better on the medical dataset. This could be attributed to the lesser number of sentences to train on for the attention model and hence it could not find appropriate weights to assign to words in a sentence.

Hence, future work could include making a bigger and more comprehensive medical dataset for paraphrase detection. Also, medical word embeddings could be used instead of word2vec or glove embeddings to represent the medical data better as many medical terms could be missing in the glove and word2vec embeddings which are trained on plain text data.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[4] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004.

[5] Natalia Grabar and Thierry Hamon. Unsupervised method for the acquisition of general language paraphrases for medical compounds. In *Proceedings of the 4th International Workshop on Computational Terminology (Computerm)*, pages 94–103, 2014.

[6] Samer Hassan and Rada Mihalcea. Semantic relatedness using salient semantic analysis. In *Aaai*. San Francisco, CA, 2011.

[7] Yangfeng Ji and Jacob Eisenstein. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896, 2013.

[8] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[9] Vasile Rus, Mihai Lintean, Rajendra Banjade, Nobal Niraula, and Dan Stefanescu. Semilar: The semantic similarity toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 163–168, 2013.

[10] Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809, 2011.

[11] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[12] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[13] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[14] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Semi-supervised clustering for short text via deep representation learning. *arXiv preprint arXiv:1602.06797*, 2016.

[15] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*, 2016.

[16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.