Paraphrase Detection using Deep Learning

A Aditya (15IT201)
Abhishek S (15IT202)
Sanjay P (15IT139)

Introduction

Paraphrase detection is the task of examining two sentences and determining whether they have the same meaning.

2 Models used : Base model and Attention based model.

Challenges

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.

No work on applying paraphrase detection to medical domain.

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. Identifying these duplicates and consolidating their answers increases the efficiency of such QA forums.

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.

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 deep learning.

Research Objectives

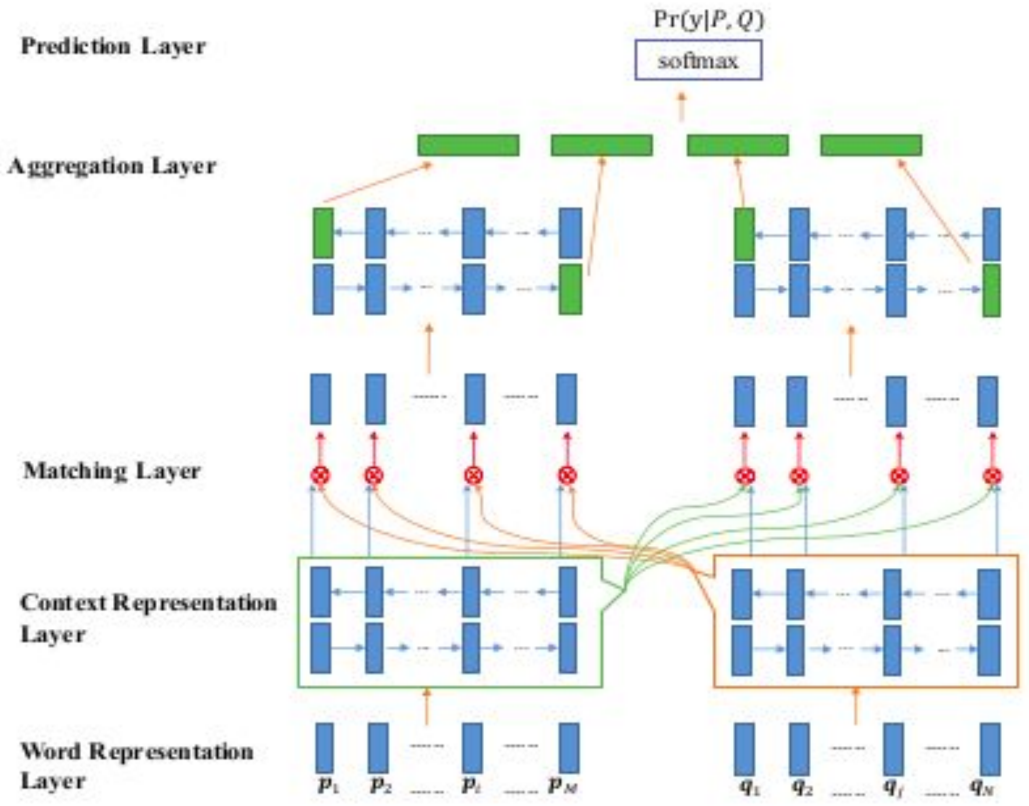
1. To implement existing models for paraphrase detection.

2. To improve existing models for paraphrase detection.

3. To develop a hybrid model for clinical paraphrase detection on clinical data.

Methodology

Base Model



Methodology

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 and word2vec embeddings.

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 bidirectional LSTM (BiLSTM) to encode contextual embeddings for each time-step of P.

Methodology

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.

$$\boldsymbol{m} = f_m(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{W}) \qquad m_k = cosine(W_k \circ \boldsymbol{v}_1, W_k \circ \boldsymbol{v}_2)$$

Full matching

In this strategy, each forward (or backward) contextual embedding is compared with the last time step of the forward (or backward) representation of the other sentence.

$$\overrightarrow{m}_i^{full} = f_m(\overrightarrow{h}_i^p, \overrightarrow{h}_N^q; W^1)$$

$$\overleftarrow{m}_i^{full} = f_m(\overleftarrow{h}_i^p, \overleftarrow{h}_1^q; W^2)$$

Max Pooling Matching

In this strategy, each forward (or backward) contextual embedding is compared with every forward (or backward) contextual embeddings of the other sentence, and only the maximum value of each dimension is retained.

$$\overrightarrow{\boldsymbol{m}}_i^{max} = \max_{j \in (1...N)} f_m(\overrightarrow{\boldsymbol{h}}_i^p, \overrightarrow{\boldsymbol{h}}_j^q; \boldsymbol{W}^3)$$

$$\overleftarrow{\boldsymbol{m}}_i^{max} = \max_{j \in (1...N)} f_m(\overleftarrow{\boldsymbol{h}}_i^p, \overleftarrow{\boldsymbol{h}}_j^q; \boldsymbol{W}^4)$$

Attentive Matching

First calculate the cosine similarities between each forward (or backward) contextual embedding and every forward (or backward) contextual embeddings of the other sentence. Then calculate an attentive vector for the entire sentence Q by weighted summing all the contextual embeddings of Q. Finally, match each forward (or backward) contextual embedding with its corresponding attentive vector:

$$\overrightarrow{\alpha}_{i,j} = cosine(\overrightarrow{\boldsymbol{h}}^p_i, \overrightarrow{\boldsymbol{h}}^q_j) \qquad j = 1, ..., N$$
$$\overleftarrow{\alpha}_{i,j} = cosine(\overleftarrow{\boldsymbol{h}}^p_i, \overleftarrow{\boldsymbol{h}}^q_j) \qquad j = 1, ..., N$$

$$\overrightarrow{\boldsymbol{h}}^{mean}_i = \frac{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j} \cdot \overrightarrow{\boldsymbol{h}}^q_j}{\sum_{j=1}^{N} \overrightarrow{\alpha}_{i,j}}$$
$$\overleftarrow{\boldsymbol{h}}^{mean}_i = \frac{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j} \cdot \overleftarrow{\boldsymbol{h}}^q_j}{\sum_{j=1}^{N} \overleftarrow{\alpha}_{i,j}}$$

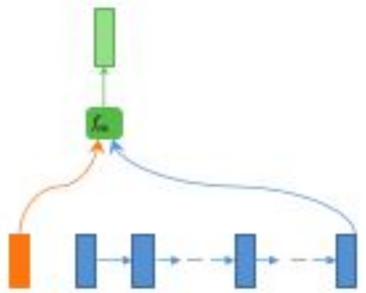
$$\overrightarrow{\boldsymbol{m}}^{att}_i = f_m(\overrightarrow{\boldsymbol{h}}^p_i, \overrightarrow{\boldsymbol{h}}^{mean}_i; \boldsymbol{W}^5)$$
$$\overleftarrow{\boldsymbol{m}}^{att}_i = f_m(\overleftarrow{\boldsymbol{h}}^p_i, \overleftarrow{\boldsymbol{h}}^{mean}_i; \boldsymbol{W}^6)$$

Max Attentive Matching

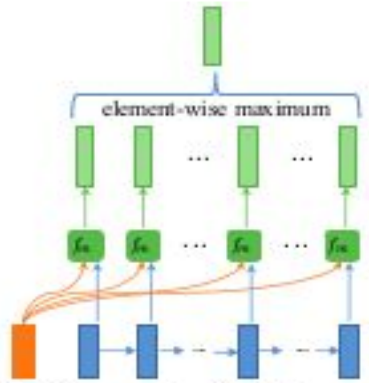
This strategy is similar to the Attentive-Matching strategy. However, instead of taking the weighted sum of all the contextual embeddings as the attentive vector, pick the contextual embedding with the highest cosine similarity as the attentive vector.

Then, match each contextual embedding of the sentence P with its new attentive vector.

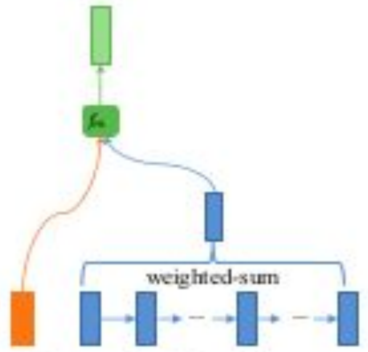
Methodology



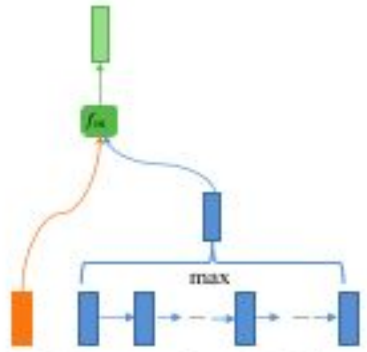
(a) Full-Matching



(b) Maxpooling-Matching



(c) Attentive-Matching



(d) Max-Attentive-Matching

Methodology

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

5. Prediction layer : The purpose of this layer is to evaluate the probability distribution Pr(y|P, Q). 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.

Methodology

Attention Model :-

Idea : Not all words contribute equally to the representation of the sentence meaning.

Extract words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector.

Model ”attends” to important parts of the sentence.

Methodology

Attention Model :-

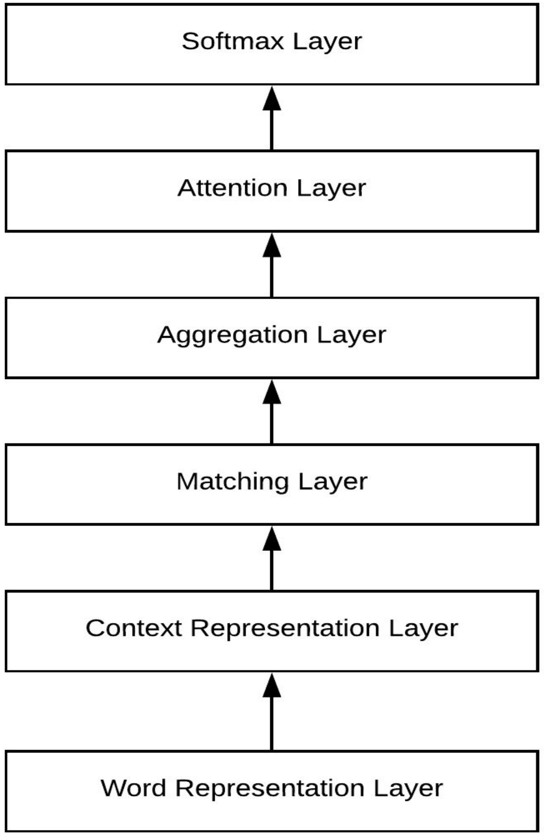
$$u_{it} = tanh(W_w h_{it} + b_w)$$

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

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

Methodology

Attention Model



Work Done

4 Tasks:

Performance comparison of LSTM and GRU cells with base model MSRP dataset.

Performance comparison of glove and word2vec embeddings with base model on MSRP dataset.

Performance comparison of Base model and Attention model on MSRP dataset.

Performance comparison of Base model and Attention model on Medical dataset.

Work Done

Medical Dataset:

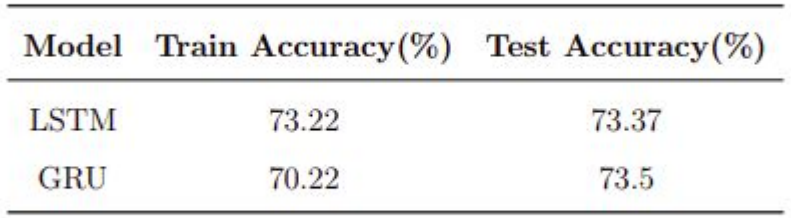
Created a medical paraphrase corpus from the clinical notes in i2b2 dataset.

Training Set : 150 pairs of sentences

Test Set : 60 pairs of sentences

Results and Analysis

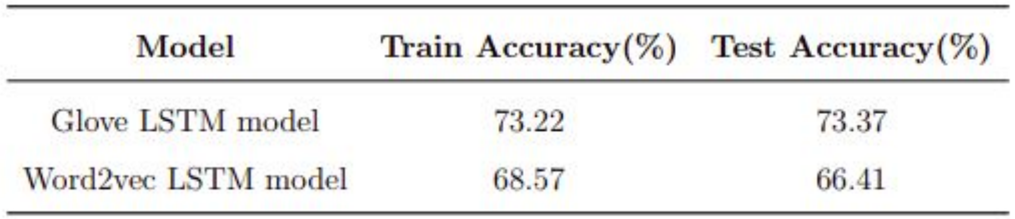
Performance comparison of LSTM and GRU cells with base model MSRP dataset.



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

Results and Analysis

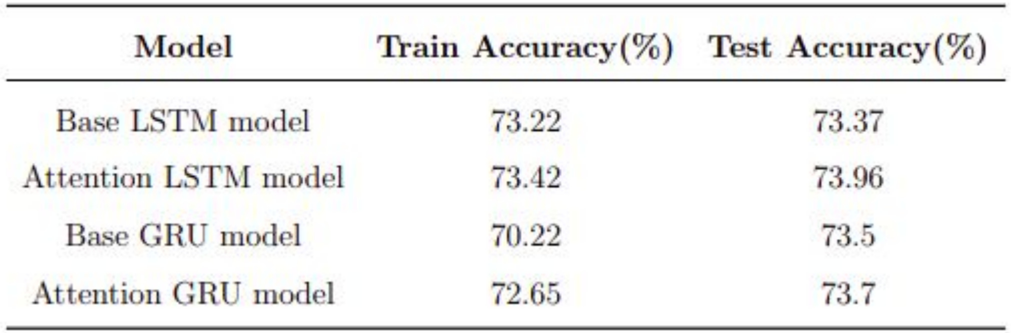
Performance comparison of glove and word2vec embeddings with base model on MSRP dataset.



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

Results and Analysis

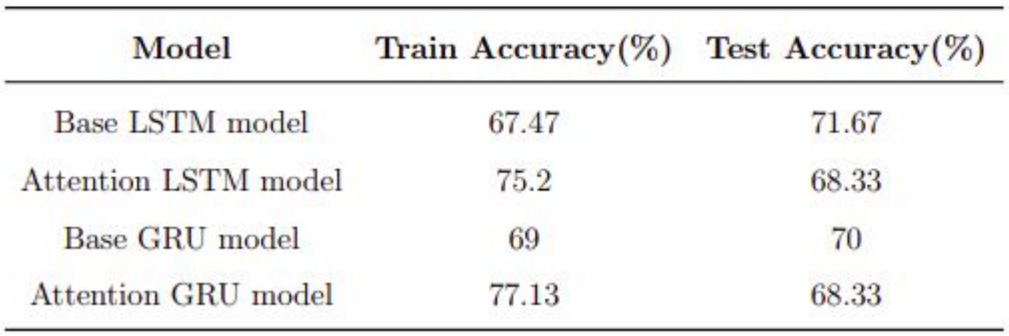
Performance comparison of Base model and Attention model on MSRP dataset.



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

Results and Analysis

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

Future Work

Making a bigger and more comprehensive medical dataset for paraphrase detection.

Medical word embeddings could be used instead of word2vec or glove embeddings.

Individual Contribution

A Aditya : Word Representation using glove pretrained vectors, Context Representation and Aggregation layers using Bidirectional LSTM, Attentive and Max - Attentive matching strategy, Attention model.

2. Abhishek S : MSRP Training and Test Dataset loader, Word Representation using word2vec pretrained vectors, Full matching strategy, Creating medical paraphrase dataset.

3. Sanjay P : Medical Training and Test dataset loader, Word Representation using word2vec pretrained vectors, Max Pooling matching strategy, Prediction layer using Softmax, Creating medical paraphrase dataset.

Thank you.