COEN 346
Quora Question Pairs Semantics Similarity Analysis

Jiahui Cao
jcao@scu.edu

Ruihao Wei
rwei@scu.edu

Abstract

In this project, we employ Natural Language Processing (NLP) techniques to analyze the sentiment similarity between question pairs. To solve the problem, we employ various NLP techniques such as word embedding, LSTM and BERT. Evaluation and analysis were given based on the performance of each model we attempted.

1 Introduction

Question-and-answer platforms have become popular in today's digital age. Quora, one of the leading question platforms, servers millions of users as their knowledge repository. Each month there are over 100 million users visit Quora and query for quality answers. It's not a surprise that many of these questions are semantics similar. Identifying these duplicated questions can provide a better experience for active seekers and writers.

This belongs to the Semantics classification NLP task. The task involves quantifying the degree of semantic similarity between two pairs of texts and identifying whether they belong to the same class or not. Therefore, we use a sequence to token neural architecture to approach the problem. In this project, we use three different architectures: Logistic Regression, Bidirectional LSTM, and BERT.

2 Background

Sentence semantics similarity is a fundamental problem in Natural Language Processing (NLP) that involves quantifying the degree of similarity or relatedness between two sentences based on their meaning. It plays a crucial role in various NLP tasks.

In general, semantic similarity is a measure of the conceptual distance between two objects, based on the correspondence of their meanings [1]. Recent work in the area of natural language processing has contributed valuable solutions to calculate the semantic similarity between words and sentences. Related works can roughly be classified into the following major categories:

- Word co-occurrence methods
- Similarity based on a lexical database
- Method based on web search engine results
- Methods based on word vectors using recursive neural networks and deep neural networks

Word co-occurrence methods are commonly used in Information Retrieval (IR) systems [2]. This method has a word list of meaningful words and every query is considered as a document. A vector is formed for the query and for documents. The relevant documents are retrieved based on the similarity between the query vector and the document vector.

Using the lexical database methodology, the similarity is computed using a predefined word hierarchy which has words, meanings, and relationships with other words compiled in a tree-like structure[3]. While comparing two words, it takes into account the path distance between the words as well as the depth of the *subsumer* in the hierarchy. The *subsumer* refers to the relative root node concerning the two words being compared. It also uses a word corpus to calculate the 'information content' of the word which influences the final similarity.

The third methodology computes relatedness based on web search engine results utilizing the number of search results [4]. This technique does not necessarily give the similarity between words as words with opposite meanings frequently occur together on the web pages which influences the final similarity index.

Recently, the models based on neural networks have produced significant improvements in the results related to semantic similarity [5]–[7]. The

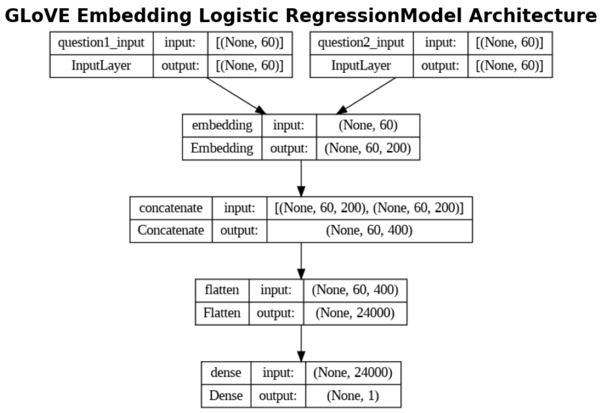

GLoVE Embedding Logistic RegressionModel Architecture
question1_input | input: | [(None, 60)]
InputLayer | output: | [(None, 60)]
question2_input | input: | [(None, 60)]
InputLayer | output: | [(None, 60)]
embedding | input: | (None, 60)
Embedding | output: | (None, 60, 200)
concatenate | input: | [(None, 60, 200), (None, 60, 200)]
Concatenate | output: | (None, 60, 400)
flatten | input: | (None, 60, 400)
Flatten | output: | (None, 24000)
dense | input: | (None, 24000)
Dense | output: | (None, 1)

Figure 1: Logsitc regression model with GLoVE embedding

approach is to utilize neural network architectures, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs), to learn sentence representations. These models can capture sequential dependencies, contextual information, and syntactic structures, enabling them to capture semantic similarity more effectively.

More recently, pre-trained language models like BERT, GPT, and RoBERTa [8]-[10]have achieved significant breakthroughs in sentence semantics similarity. These models are trained on large amounts of text data and can generate contextualized representations of sentences, capturing complex semantic relationships and contextual nuances.

3 Approach

3.1 Logistic Regression

The first architecture we use is the classical Logistic Regression combined with the GLoVE embedding. The GLoVE embedding layer converts a sequence of text input from a word to a GLoVE vector representation. Then two sequences of representation are concatenated together and get flattened. Finally, we use the concatenated GLoVE embedding sequence as an input to the classical logistic regression model with Sigmoid as our activation function.

let e^1 denote GloVE embedding for input sequence 1 and e^2 denote GloVE embedding for input sequence 2, we have the concatenated embedding

$$e = \{e^1; e^2\} \tag{1}$$

The hidden layer h is

$$h = w^T e + b \tag{2}$$

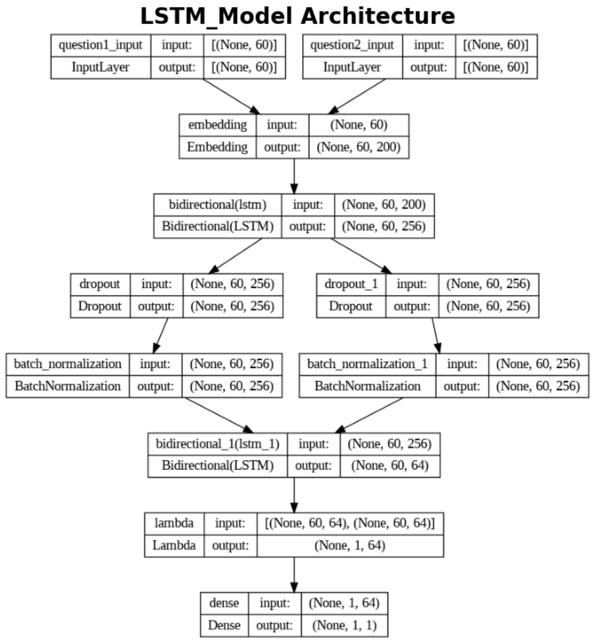

LSTM_Model Architecture
question1_input | input: | [(None, 60)]
InputLayer | output: | [(None, 60)]
question2_input | input: | [(None, 60)]
InputLayer | output: | [(None, 60)]
embedding | input: | (None, 60)
Embedding | output: | (None, 60, 200)
bidirectional(lstm) | input: | (None, 60, 200)
Bidirectional(LSTM) | output: | (None, 60, 256)
dropout | input: | (None, 60, 256)
Dropout | output: | (None, 60, 256)
dropout_1 | input: | (None, 60, 256)
Dropout | output: | (None, 60, 256)
batch_normalization | input: | (None, 60, 256)
BatchNormalization | output: | (None, 60, 256)
batch_normalization_1 | input: | (None, 60, 256)
BatchNormalization | output: | (None, 60, 256)
bidirectional_1(lstm_1) | input: | (None, 60, 256)
Bidirectional(LSTM) | output: | (None, 60, 64)
lambda | input: | [(None, 60, 64), (None, 60, 64)]
Lambda | output: | (None, 1, 64)
dense | input: | (None, 1, 64)
Dense | output: | (None, 1, 1)

Figure 2: LSTM Siamese network with GloVE embedding

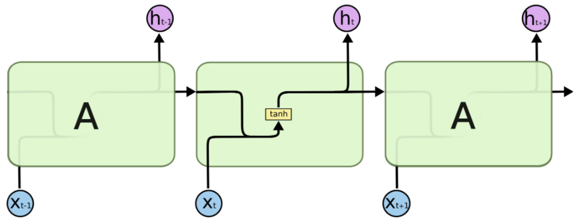


Figure 3: RNN block [11]

where W is the weight and b is the bias.

Then we get the probability output by using a sigmoid function

$$p = \sigma(h) = \frac{1}{1 + exp^{-h}} \tag{3}$$

The loss we use to train our model is the binary cross entropy

$$L = -(y \log(p) + (1 - y) \log(1 - p)) \tag{4}$$

3.2 Bidirectional LSTM

The second architecture we use is a Siamese network with LSTM and GLoVE embedding.

Long Short-Term Memory Networks, or "LSTM", are a special kind of RNN, capable of learning long-term dependencies. LSTMs have a chain-like structure the same as RNN, but the repeating module has a different structure. Instead of having a simple $tanh$ layer, LSTMs have four interacting layers. The core idea behind LSTMs is

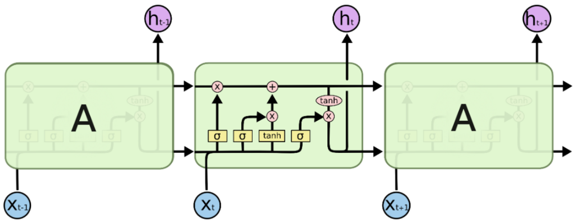


Figure 4: LSTM block [11]

the cell state, the horizontal line running through the top of Figure 4. And LSTM has the ability to remove or add information to the cell state, carefully regulated by four interacting layers. We are not going into the detail of these interacting layers because we didn't really build the LSTM in this project but rather use it as a building block. Generally, we have a cell state C_t and a hidden state h_t for each LSTM block, and they're controlled by the four gate layers. The hidden state h_t is decided by the cell state C_t

$$h_t = o_t * tanh(C_t) \tag{5}$$

where o_t is decided by the previous hidden state h_{t-1} and the current input x_t

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{6}$$

which is similar to the logistic regression[11].

Another building block we use in this approach is the Siamese network. A Siamese Neural Network is a class of neural network architectures that contain two or more identical subnetworks. It is used to find the similarity of the inputs by comparing their feature vectors. Traditionally, it uses a distance-based loss. The loss is used to learn embedding in which two similar points have a low Euclidean distance and two dissimilar points have a large Euclidean distance[12]. In this approach, we didn't use a distance-based loss, instead, we modified the Siamese network in order to keep using the binary cross-entropy.

The modification we did is adding a distance layer. Instead of train on a distance-based loss, we calculated the distance between two feature vectors and use the distance as input to the output layer, which is a regular sigmoid output layer to get a probability. The distance we used is the Manhattan distance

$$||h^{left} - h^{right}||_1 \tag{7}$$

Let's put all our building blocks together. Two sequence inputs are first passed through the GloVE embedding layer to get GloVE embedding. Then, they pass through two identical 2-layer Bidirectional LSTM subnetworks. We use the bidirectional LSTM because we have the full sequence as input. And we also add batch normalization and drop-out layers between 2 LSTM layers. Then we calculate the Manhattan distance and get the probability output through an output layer. The loss we use is still the binary cross entropy.

3.3 Sentence-BERT

The third approach we employed is Sentence-BERT, an adaptation of the pre-trained BERT network that utilizes siamese and triplet network architectures to generate semantically meaningful sentence embeddings. These embeddings can be compared using cosine similarity, thereby reducing the computational effort required to find the most similar pair compared to BERT/RoBERTa in a matter of seconds. Despite this efficiency gain, Sentence-BERT maintains the accuracy achieved by BERT in capturing semantic similarity.

The Siamese neural networks as talked about before are the type of neural network architecture designed for learning similarity or dissimilarity between pairs of inputs. Instead of implementing it with an LSTM model, here we use the pre-trained BERT model to generate siamese sentences directly and then put the results as input to our LSTM models. The aim of doing this is to compare the results of this approach with the second approach we implemented. To see if the BERT model will bring improvements to the results.

4 Experiments

4.1 Data Preprocessing

The dataset we used in our project is the Quora Question pair dataset. It has the following fields:

- id - the id of a training set question pair
- qid1, qid2 - unique ids of each question (only available in train.csv)
- question1, question2 - the full text of each question
- is duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning and 0 otherwise.

After dropping some bad data, we translated all words to lowercase and drop symbols. Then, we

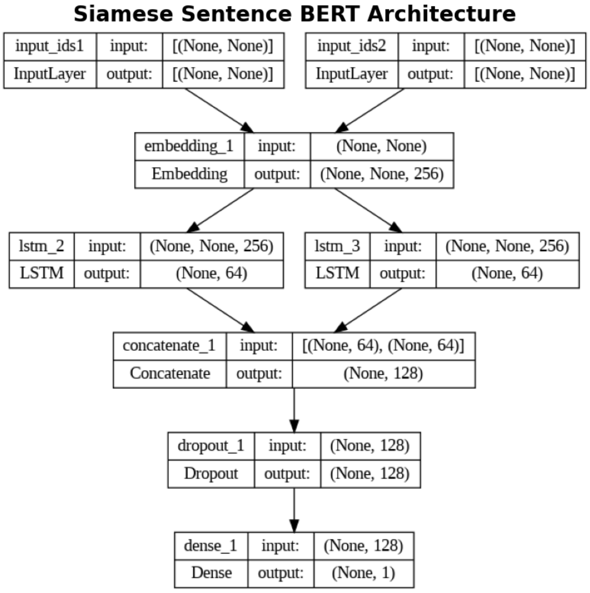
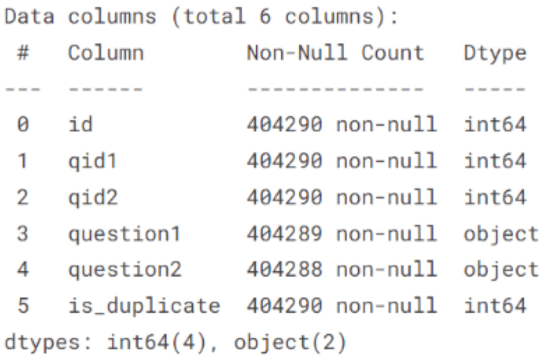
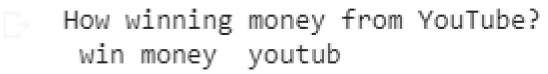

Siamese Sentence BERT Architecture
input_ids1 | input: | [(None, None)]
InputLayer | output: | [(None, None)]
input_ids2 | input: | [(None, None)]
InputLayer | output: | [(None, None)]
embedding_1 | input: | (None, None)
Embedding | output: | (None, None, 256)
lstm_2 | input: | (None, None, 256)
LSTM | output: | (None, 64)
lstm_3 | input: | (None, None, 256)
LSTM | output: | (None, 64)
concatenate_1 | input: | [(None, 64), (None, 64)]
Concatenate | output: | (None, 128)
dropout_1 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 1)

Figure 5: Siamese Sentence BERT Architecture



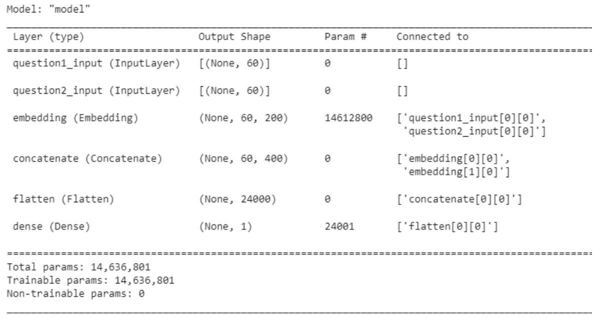
```
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            404290 non-null  int64
 1   qid1          404290 non-null  int64
 2   qid2          404290 non-null  int64
 3   question1     404289 non-null  object
 4   question2     404288 non-null  object
 5   is_duplicate  404290 non-null  int64
dtypes: int64(4), object(2)
```

Figure 6: The dataset we used in our project



```
How winning money from YouTube?
 win money  youtub
```

Figure 7: Example of pre-processed and post-processed data



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 question1_input (InputLayer)   [(None, 60)]         0           []

 question2_input (InputLayer)   [(None, 60)]         0           []

 embedding (Embedding)          (None, 60, 200)      14612800    ['question1_input[0][0]',
                                                                  'question2_input[0][0]']

 concatenate (Concatenate)      (None, 60, 400)      0           ['embedding[0][0]',
                                                                  'embedding[1][0]']

 flatten (Flatten)              (None, 24000)        0           ['concatenate[0][0]']

 dense (Dense)                  (None, 1)            24001       ['flatten[0][0]']

==================================================================================================
Total params: 14,636,801
Trainable params: 14,636,801
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 8: A detailed configuration of logistic regression model

applied stemmer to reduce all words to their base English form and drop the stopwords. Finally, we applied tokenization and sequence padding to the data.

4.2 Logistic Regression

4.2.1 Configurations

- GloVE embedding dimension - 200
- Sequence input length - 60
- Concataned embedding layer dimension: 60, 400
- Batch size - 2048
- Epoch - 30

4.2.2 Evaluation metrics

Metrics	Value
Accuracy	0.735
Loss	0.549
Precision	0.615
Recall	0.759
F1	0.678

Our classical logistic regression model reached 73.5% accuracy on validation data with a $F1$ score 0.678. Although we have an acceptable accuracy rate, we can see from the relatively low precision that the model is overfitted to guess 1.

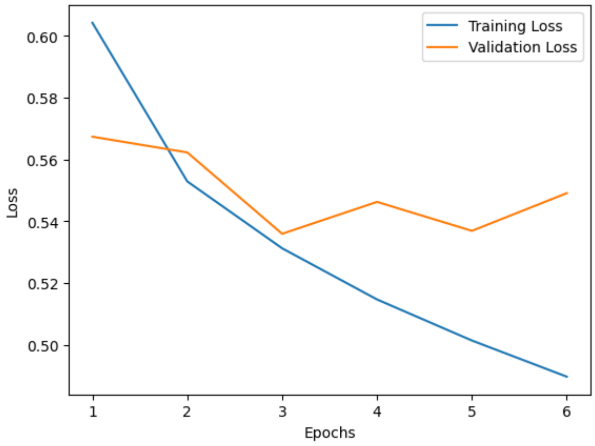


Figure 9: The loss history of logistic regression model

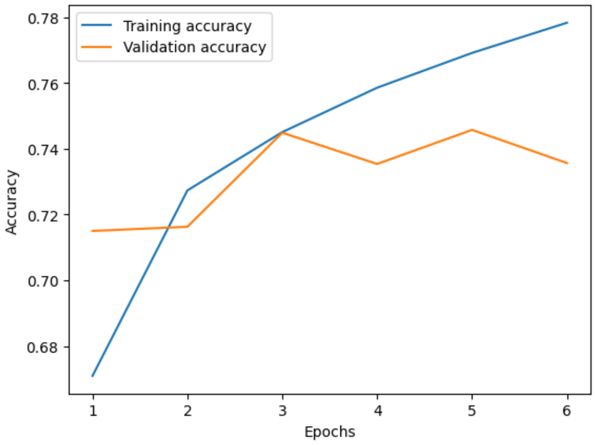


Figure 10: The accuracy history of logistic regression model

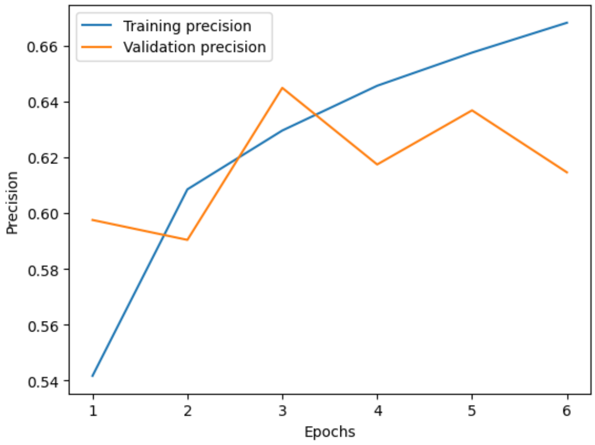


Figure 11: The precision history of logistic regression model

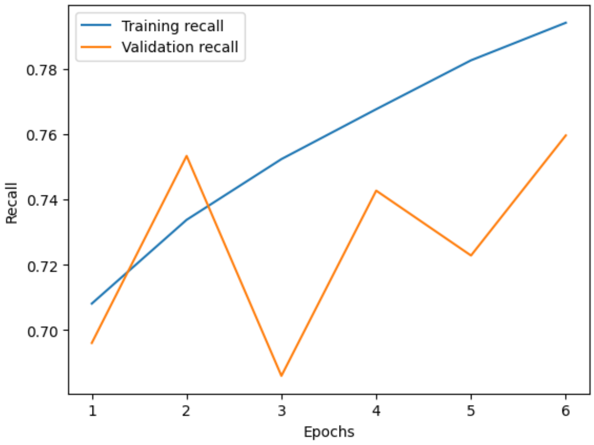
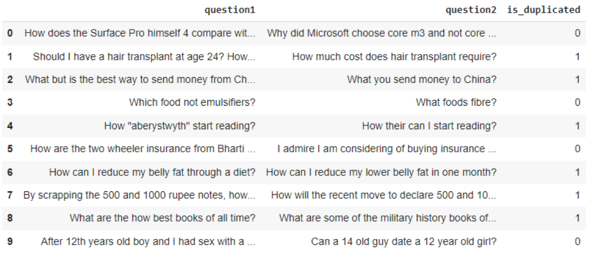


Figure 12: The recall history of logistic regression model

4.2.3 example results



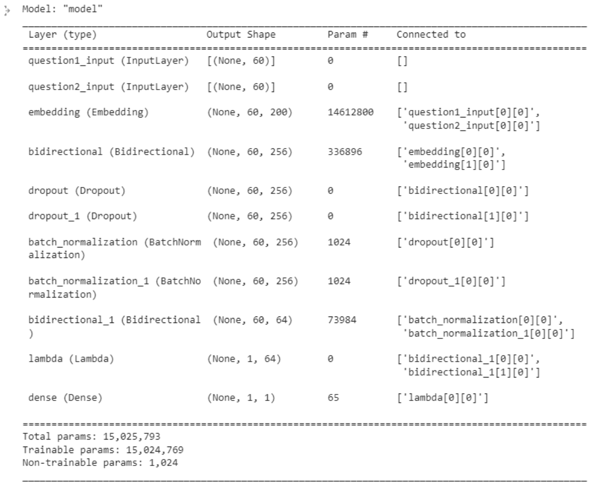
	question1	question2	is_duplicated
0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...	0
1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?	1
2	What but is the best way to send money from Ch...	What you send money to China?	1
3	Which food not emulsifiers?	What foods fibre?	0
4	How "aberystwyth" start reading?	How their can I start reading?	1
5	How are the two wheeler insurance from Bharti ...	I admire I am considering of buying insurance ...	0
6	How can I reduce my belly fat through a diet?	How can I reduce my lower belly fat in one month?	1
7	By scrapping the 500 and 1000 rupee notes, how...	How will the recent move to declare 500 and 10...	1
8	What are the how best books of all time?	What are some of the military history books of...	1
9	After 12th years old boy and I had sex with a ...	Can a 14 old guy date a 12 year old girl?	0

Figure 13: 10 example runs of logistic regression model prediction

4.3 LSTM

4.3.1 Configurations

- GloVE embedding dimension - 200
- Sequence input length - 60
- Bidirectional LSTM layer 1 dimension: 60, 256
- Batch normalization dimension: 60, 256
- Drop out dimension: 60, 256
- Bidirectional LSTM layer 2 dimension: 60, 64
- Batch size - 2048
- Epoch - 30



```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 question1_input (InputLayer)   [(None, 60)]         0           []

 question2_input (InputLayer)   [(None, 60)]         0           []

 embedding (Embedding)          (None, 60, 200)      14612800    ['question1_input[0][0]',
                                                                  'question2_input[0][0]']

 bidirectional (Bidirectional)  (None, 60, 256)      336896      ['embedding[0][0]',
                                                                  'embedding[1][0]']

 dropout (Dropout)              (None, 60, 256)      0           ['bidirectional[0][0]']

 dropout_1 (Dropout)            (None, 60, 256)      0           ['bidirectional[1][0]']

 batch_normalization (BatchNorm  (None, 60, 256)     1024        ['dropout[0][0]']
 alization)

 batch_normalization_1 (BatchNo  (None, 60, 256)     1024        ['dropout_1[0][0]']
 rmalization)

 bidirectional_1 (Bidirectional  (None, 60, 64)      73984       ['batch_normalization[0][0]',
 )                                                                'batch_normalization_1[0][0]']

 lambda (Lambda)                (None, 1, 64)        0           ['bidirectional_1[0][0]',
                                                                  'bidirectional_1[1][0]']

 dense (Dense)                  (None, 1, 1)         65          ['lambda[0][0]']

==================================================================================================
Total params: 15,025,793
Trainable params: 15,024,769
Non-trainable params: 1,024
__________________________________________________________________________________________________
```

Figure 14: A detailed configuration of Bidirectional LSTM model

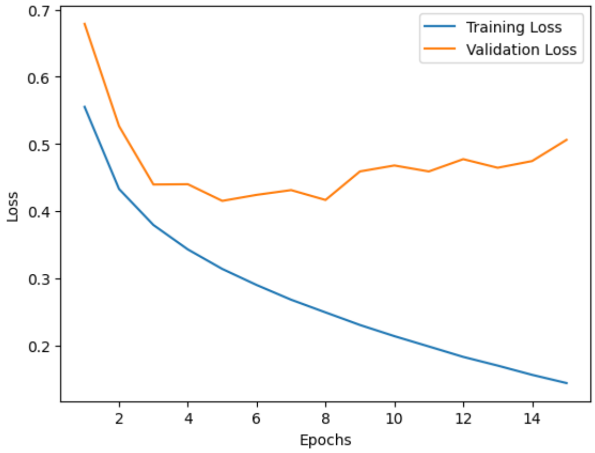


Figure 15: The loss history of LSTM model

4.3.2 Evaluation metrics

Metrics	Value
Accuracy	0.832
Loss	0.474
Precision	0.759
Recall	0.798
F1	0.778

Our bidirectional LSAM model reached 83.5% accuracy on validation data with a $F1$ score of 0.778. It performs much better than the logistic regression model

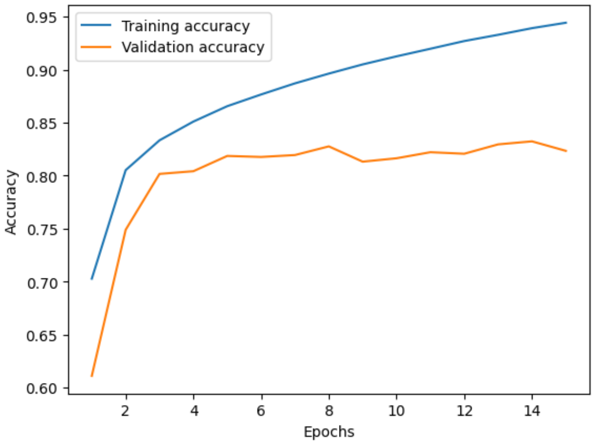


Figure 16: The accuracy history of LSTM model

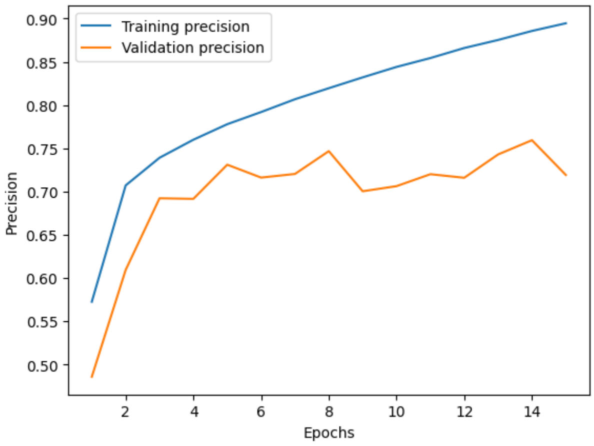


Figure 17: The precision history of LSTM model

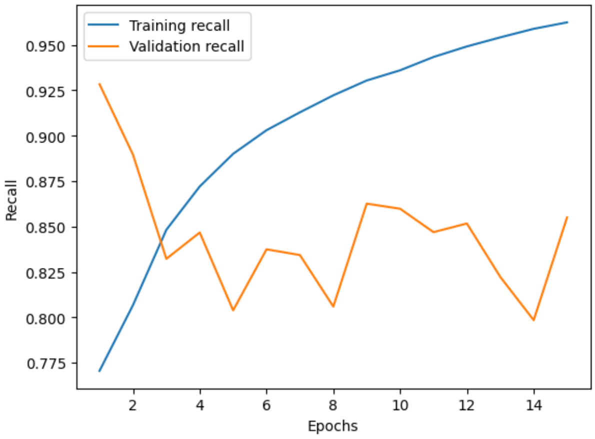
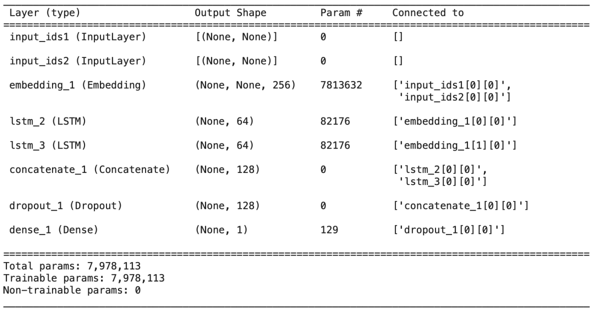


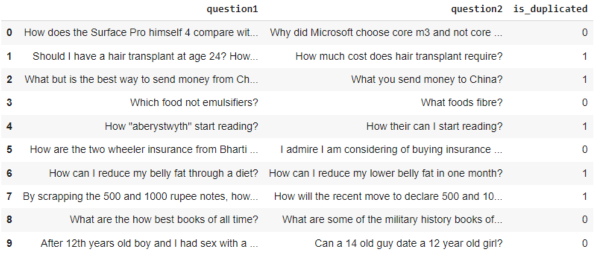
Figure 18: The recall history of LSTM model



```
Layer (type)                 Output Shape         Param #     Connected to
==================================================================================================
input_ids1 (InputLayer)      [(None, None)]       0           []

input_ids2 (InputLayer)      [(None, None)]       0           []

embedding_1 (Embedding)      (None, None, 256)    7813632     ['input_ids1[0][0]',
                                                               'input_ids2[0][0]']

lstm_2 (LSTM)                (None, 64)           82176       ['embedding_1[0][0]']

lstm_3 (LSTM)                (None, 64)           82176       ['embedding_1[1][0]']

concatenate_1 (Concatenate)  (None, 128)          0           ['lstm_2[0][0]',
                                                               'lstm_3[0][0]']

dropout_1 (Dropout)          (None, 128)          0           ['concatenate_1[0][0]']

dense_1 (Dense)              (None, 1)            129         ['dropout_1[0][0]']

==================================================================================================
Total params: 7,978,113
Trainable params: 7,978,113
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 20: A detailed configuration of Siamese Sentence-BERT model

4.3.3 example results



	question1	question2	is_duplicated
0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...	0
1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?	1
2	What but is the best way to send money from Ch...	What you send money to China?	1
3	Which food not emulsifiers?	What foods fibre?	0
4	How "aberystwyth" start reading?	How their can I start reading?	1
5	How are the two wheeler insurance from Bharti ...	I admire I am considering of buying insurance ...	0
6	How can I reduce my belly fat through a diet?	How can I reduce my lower belly fat in one month?	1
7	By scrapping the 500 and 1000 rupee notes, how...	How will the recent move to declare 500 and 10...	1
8	What are the how best books of all time?	What are some of the military history books of...	0
9	After 12th years old boy and I had sex with a ...	Can a 14 old guy date a 12 year old girl?	0

Figure 19: 10 example runs of LSTM model prediction

4.4 BERT

4.4.1 Configurations

- Sentence BERT embedding dimension: 200
- Sequence input length: 60
- Bidirectional LSTM layer 1 dimension: 256
- Bidirectional LSTM layer 2 dimension: 64
- Concatenate dimension: 128
- Dropout dimension: 128
- Batch size: 2048
- Epoch: 30

4.4.2 Evaluation metrics

Metrics	Value
Accuracy	0.795
Loss	0.462
Precision	0.747
Recall	0.541
F1	0.627

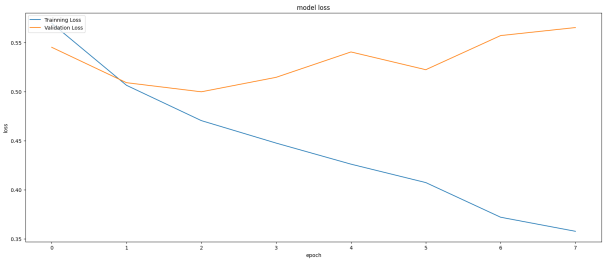


Figure 21: The loss history of BERT model

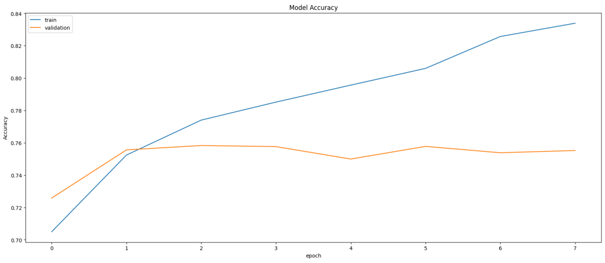


Figure 22: The accuracy history of BERT model

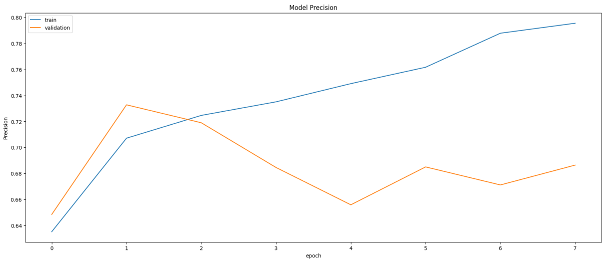


Figure 23: The precision history of BERT model

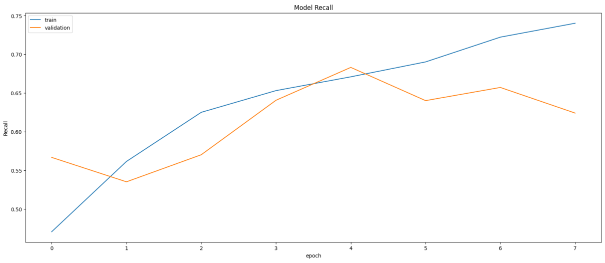
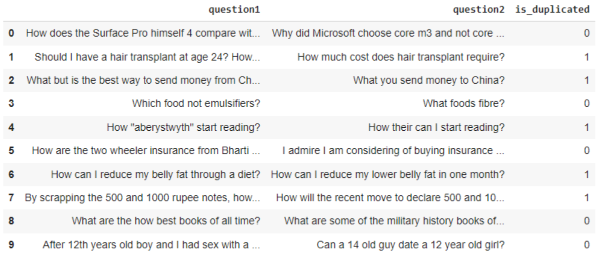


Figure 24: The recall history of BERT model

4.4.3 example results



	question1	question2	is_duplicated
0	How does the Surface Pro himself 4 compare wit...	Why did Microsoft choose core m3 and not core ...	0
1	Should I have a hair transplant at age 24? How...	How much cost does hair transplant require?	1
2	What but is the best way to send money from Ch...	What you send money to China?	1
3	Which food not emulsifiers?	What foods fibre?	0
4	How "aberystwyth" start reading?	How their can I start reading?	1
5	How are the two wheeler insurance from Bharti ...	I admire I am considering of buying insurance ...	0
6	How can I reduce my belly fat through a diet?	How can I reduce my lower belly fat in one month?	1
7	By scrapping the 500 and 1000 rupee notes, how...	How will the recent move to declare 500 and 10...	1
8	What are the how best books of all time?	What are some of the military history books of...	0
9	After 12th years old boy and I had sex with a ...	Can a 14 old guy date a 12 year old girl?	0

Figure 25: 10 example runs of BERT model prediction

5 Conclusion and future improvement

Based on the results obtained from logistic regression, LSTM, and BERT, we can draw the following conclusions.

The logistic regression model achieved moderate performance with an accuracy of 73.5% indicating that the model is able to predict duplicate pairs kind of correctly. However, the F1 score (0.67) suggests that the model still has space for improvement.

The Bidirectional LSAM model achieved the best performance with an accuracy of 83.2%. It also has the highest F1 score(0.77). Despite the simple idea behind the model: predicting the semantics similarity based on token distance in vector space, it is our best model. A simple model doesn't always mean worse performance.

The Siamese Sentence-BERT model achieved competitive accuracy and loss values. However, compared with the LSTM model, the accuracy of this model is lower. In my opinion, the results might be related to fine-tuning, since BERT models usually require fine-tuning on the specific task at hand to adapt them to the data set. Fine-tuning helps the model learn task-specific features and can significantly improve performance.

Here are some potential future improvements we can make to improve our models

- Refine the model architecture by tuning the hyper-parameters we used in our training process.
- Develop an anti-noise strategy to deal with outliers in our data set.
- Improve the fine-tuning procedures of BERT model to achieve higher accuracy of the existing model
- Create new data using data augmentation swapping words from the text with synonyms.
- Train our own word embeddings on the data questions instead of using pre-trained GloVE embedding so we will have all the tokens represented in the vector space.

References

[1] D.Lin, "An information theoretic definition of similarity," inProc.ICML, vol. 98. 1998, pp. 296–304.

[2] C.T.Meadow,Text Information Retrieval Systems. NewYork, NY, USA: Academic, 1992.

[3] Y. Li, D. McLean, Z. A. Bandar, and J. D. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," IEEE Trans. Knowl. Data Eng., vol. 18, no. 8, pp. 1138–1150, Aug. 2006.

[4] D. Bollegala, Y. Matsuo, and M. Ishizuka, "Measuring semantic similarity between words using Web search engines," in Proc. WWW, vol. 7, 2007, pp. 757–766.

[5] Z. He, S. Gao, L. Xiao, D. Liu, H. He, and D. Barber, "Wider and deeper, cheaper and faster: Tensorized LSTMS for sequence learning," in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1–11.

[6] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning- ing sentence similarity," in Proc. AAAI, 2016, pp. 2786–2792.

[7] K. S. Tai, R. Socher, and C. D. Manning. (2015). "Improved semantic representations from tree-structured long short-term memory networks." [Online]. Available: https://arxiv.org/abs/1503.00075

[8] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners. OpenAI Blog, 1(8).

[10] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.

[11] C. Olah, "Understanding LSTM networks," Understanding LSTM Networks – colah's blog, 27-Aug-2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 14-Jun-2023]

[12] Benhur, S. (2022, June 5). A friendly introduction to Siamese networks. Medium. https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942