CS224N Project: Natural Language Inference for Quora Dataset

Kuy Hun Koh Yoo
Energy Resources Engineering
Stanford University
Stanford, CA 94305
kohykh@stanford.edu

Muhammad M. Almajid
Energy Resources Engineering
Stanford University
Stanford, CA 94305
majimm0a@stanford.edu

Zhi Yang Wong
Energy Resources Engineering
Stanford University
Stanford, CA 94305
zhiyangw@stanford.edu

Abstract

Attention-based Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) cells were applied to identify duplicate question-pairs in the Quora dataset. The implementation of this architecture as well as other neural architectures is described in this project. The optimal architecture was then chosen based on the F1 score. Hyper-parameter tuning was then performed to maximize the accuracy of the model. The final test score obtained was 0.82 with an F1 score of 0.76.

1 Introduction and Problem Statement

A key element for question and answer website efficiency such as Quora relies on properly categorizing questions such no questions with identical intent are duplicated. This allows users to both find and answer questions easily without having to search through all the different duplicate question pages. For example, the questions "How do you start a bakery?" and "How can one start a bakery business?" should not have separate pages because the intent of both questions are identical. This problem makes it imperative for us to recognize entailment relations between the pairs of questions using neural network architectures. Long short-term memory recurrent neural networks have been shown to be successful to identify entailment relations between sentences [4]. In this project, we used natural language processing and deep learning algorithms to identify semantically equivalent questions. Specifically, we used an LSTM-based RNN with attention to reach our goal.

2 Dataset

2.1 Description

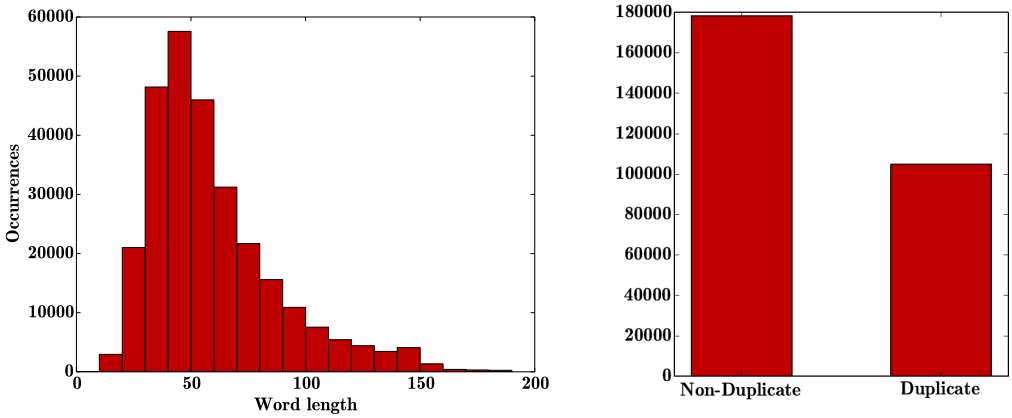
We used the dataset released by Quora [1] that consists of over 400,000 lines of potential duplicate pairs. It is important to note that the dataset was supplemented with negative examples to balance

the number of positive and negative examples. Figure 1 shows the format of the raw dataset. That is for each pair of questions we have a sample ID, an individual question ID, the questions and their corresponding labels (duplicate = 1, not duplicate = 0).

id	qid1	qid2	question1	question2	label
20	41	42	Why do rockets look white?	Why are rockets and boosters painted white?	1
21	43	44	What's causing someone to be jealous?	What can I do to avoid being jealous of someone?	0
22	45	46	What are the questions should not ask on Quora?	Which question should I ask on Quora?	0

Figure 1: Sample lines of the data set

The distribution of the number of words per question is shown in Figure 2a. It can be seen that the distribution is left skewed and has a mean of 60, standard deviation of 30. These values could be important in deciding the minimum sequence length such that we capture the semantic of a question while at the same time maximizing computational efficiency. Further, qualitative exploration of the dataset showed that the main idea of a question was captured in the first or last words of the sentence. This was especially true for the longer sentences. This hypothesis was tested and confirmed during hyper-parameter tuning.



(a) Word Length Distribution (b) Label Count in Train Data

Figure 2: Input data statistics

2.2 Pre-Processing

During tokenization, all punctuations were separated from words to achieve better accuracy. Common words (stopwords), however, were not removed from the questions before training. The data was split randomly into 70% training, 15% development and 15% test datasets. The training data had about 63% non-duplicate examples and about 37% duplicate examples as shown in Figure 2b. Given the imbalance in the dataset, both the accuracy and the F1 score for the development and test datasets are reported.

3 Methodology/Algorithm

3.1 Background

In order to establish a starting point for this project, the latest research on language RNN models were reviewed [2, 4]. The attention mechanism has been shown to improve neural models by focusing on specific words in a sentence during training [2]. This approach seemed adequate for our

purposes as it gives a way to compare the semantics of each question pair, ensuring that only the words that are relevant are taken into account.

A baseline of the effectiveness of our model was obtained from the results posted on the recently released Kaggle competition results. Among the best implementations to date (March 21, 2017) is an LSTM RNN architecture constructed by Quora that has an accuracy of $\sim$86%. It has been reported in social media that human accuracy on this dataset is also $\sim$86%.

3.2 Approach

This problem was approached by proposing and constructing several RNN architectures which were ranked based on their development accuracy with a set of fixed hyper-parameters. The architectures were designed in increasing complexity based on what seemed reasonable to best tackle the problem at hand. The best model was then tuned by testing random combinations of hyper-parameters. Due to the wide range of the number of words in a question, a maximum sequence length was selected to be hyper-parameter. The mean question length was initially used in the preliminary testing, but later hyper-parameter tuning showed that the first 20 words of each question capture its essential semantics.

3.3 Proposed Neural Network Architectures

Following are descriptions of the RNN architectures that were tested in this project.

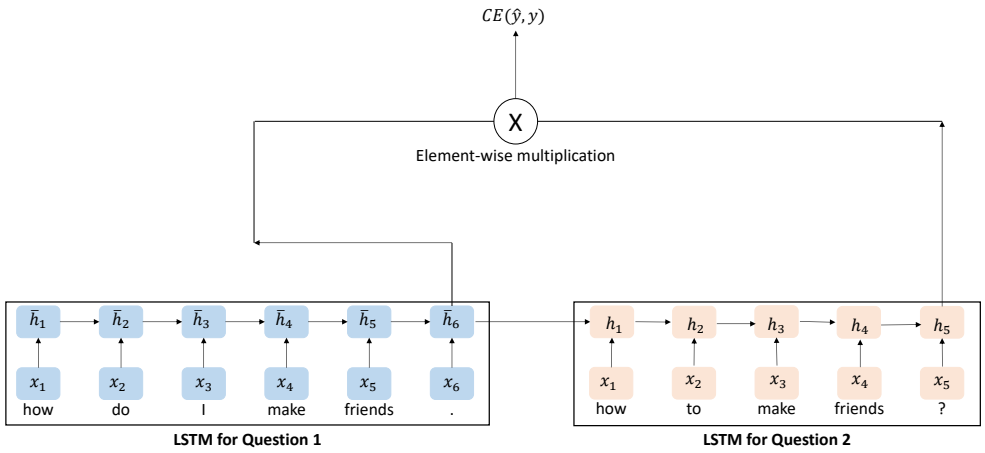


Figure 3: Architecture 1 :The first naive approach considered two LSTM RNNs to parse the pair of questions. The final hidden states of each LSTM are combined by an element-wise multiplication. The loss is computed by the cross-entropy function.

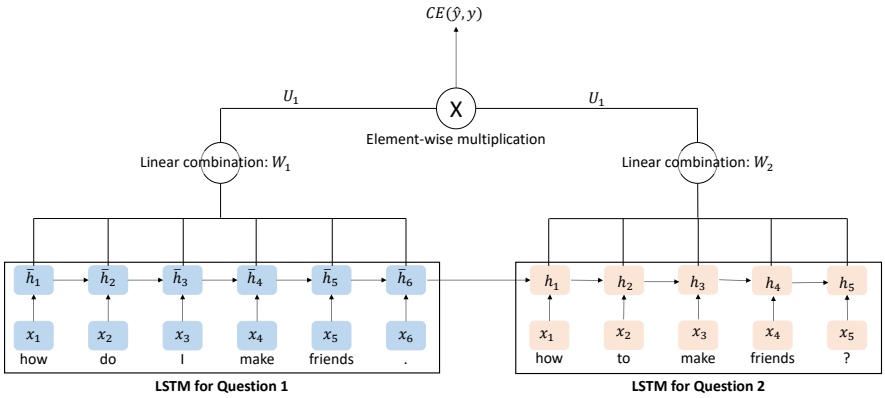


Figure 4: Architecture 2: Similar to architecture 1, two LSTM cells were considered. The hidden states of each RNN were combined by a linear combination, where the weights were trained parameters. A linear transformation was applied to each LSTM output and the similarity between the pair of questions was determined by element-wise multiplication. The loss was then computed by cross-entropy function. This architecture was tested using the questions in the forward and reverse directions.

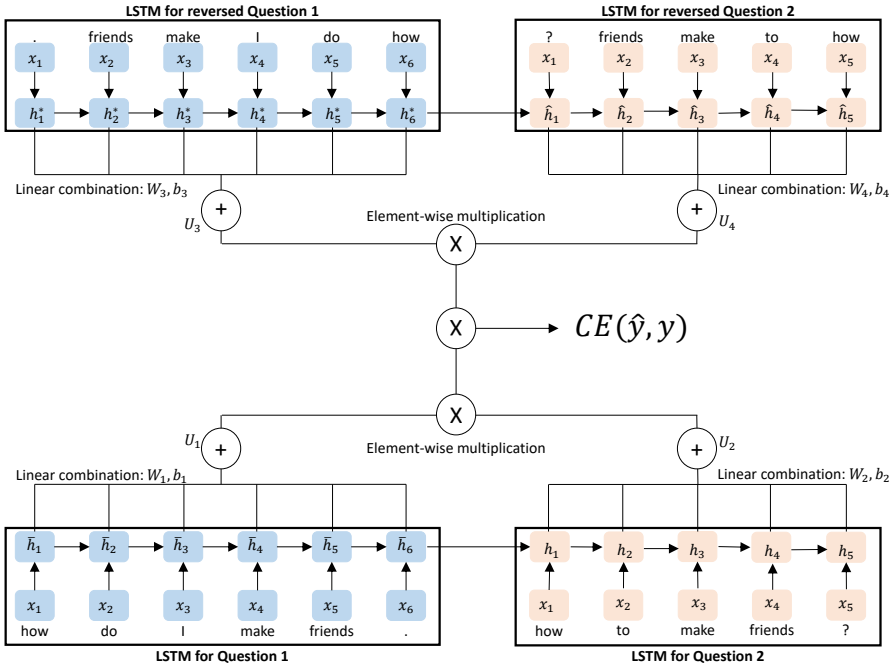


Figure 5: Architecture 3: In an attempt to replicate a bi-directional RNN, architecture 2 was duplicated to take the same pair of questions in the forward and reverse directions. The outputs were combined by an element-wise multiplication.The loss was then computed by cross-entropy function.

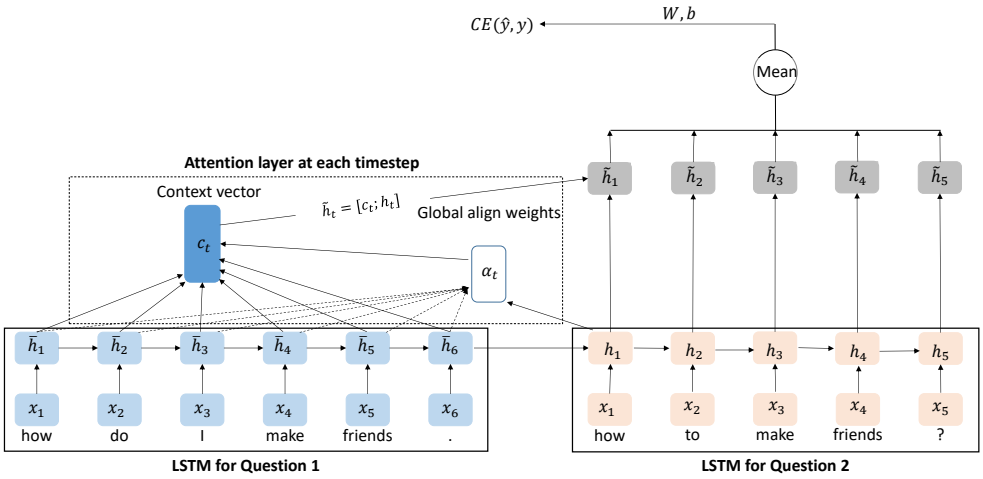


Figure 6: Architecture 4: This architecture is an attempt to formally implement global attention as proposed by [2]. The final hidden layer output represents a concatenation of each hidden state vector from the LSTM corresponding to question two with its corresponding context vector. This context vector is calculated by taking the weighted sum of the hidden output from the first question. These weights were generated from a bilinear score function. This is then combined by taking the mean of the vectors. The loss was then computed by cross-entropy function. It is noted that the bilinear score function was found to perform better than the dot product or the concatenation of the two hidden states of the two questions. Unfortunately, the various score functions were only tested on this architecture and not on our final neural architecture.

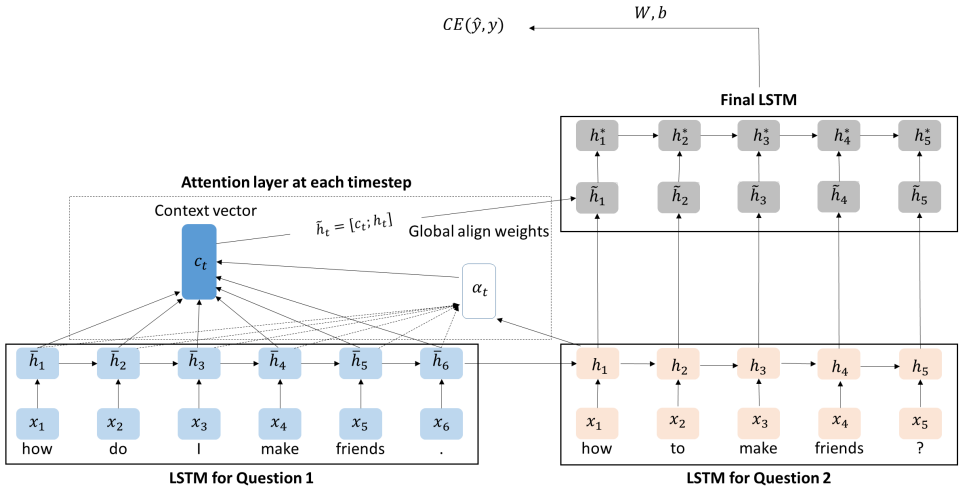


Figure 7: Final Architecture: This is a replication of architecture 4 with the difference being that the output from the attention layer is passed on to a third LSTM cell. The loss is then computed by cross-entropy function of the last output of third LSTM cell. This architecture is denoted as the "Final" as it gave the best development accuracy.

Table 1 summarizes the overall performance of the different architectures described above. An attempt was made to keep all hyper-parameters constant, but a few examples were computationally constrained due to memory issues. For example, the Final Architecture had to be tested with smaller hidden size and smaller max word length. It should also be noted that the dropout rate was initially chosen to be 0.5 and was later found to be unusually large for RNNs. This could explain the similarity in development scores for all the different models regardless of their complexity. Unfortunately, these examples were not re-tested due to time constraints.

Table 1: Hyper-parameters used for different architectures and corresponding development accuracy

Architecture	Order	GloVe	Dropout	Hidden Size	Embed Size	Max Length	Cell	Number Epochs	Dev Accuracy
Final	Normal	6B Tokens	0.2	50	50	20	LSTM	5	0.811
1	Normal	6B Tokens	0.5	150	50	60	LSTM	5	0.770
2	Normal	6B Tokens	0.5	150	50	60	LSTM	10	0.770
2	Reverse	6B Tokens	0.5	150	50	60	LSTM	5	0.770
3	Bidirectional	6B Tokens	0.5	150	50	60	LSTM	5	0.770
4	Normal	6B Tokens	0.2	100	100	60	LSTM	2	0.781

4 Results

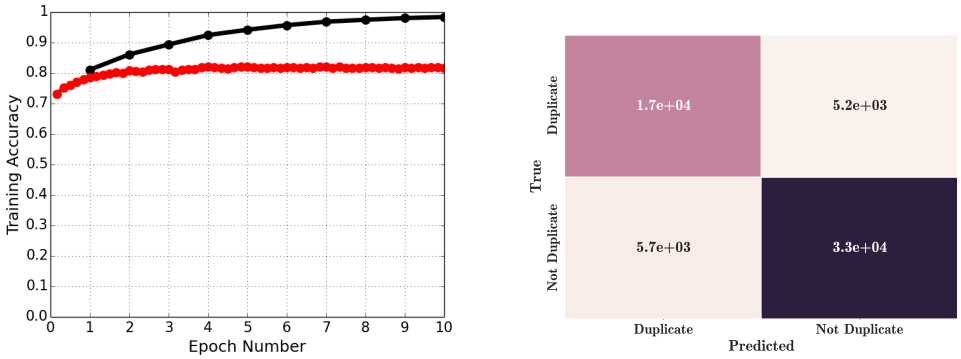
4.1 Neural Network Architecture Performance

Once the final neural architecture was identified, hyper-parameter tuning was performed to optimize the model's performance. Architecture 5 was chosen to be the base architecture. The base parameters for our model are found in the first row of Table 2. The best development accuracy and F1 score obtained were 0.821 and 0.761, respectively. Using the test data set, an accuracy of 0.820 and an F1 score of 0.759 were achieved.

Table 2: Trained Parameters

Dropout	0.20
Hidden Size	50
Embedding Size	50
Optimizer	Adam
Learning Rate	0.001
Batch Size	50
Max Length	20

The training and development performance for this architecture is shown in Figure 8a. After about 4-5 epochs, our model's accuracy for the development set plateaus. Although not shown, this stagnant behavior is consistent for the results of other hyper-parameters. The confusion matrix (Figure 8b) for the base architecture is diagonally dominant which further supports that the constructed model is predicting the labels accurately. The confusion matrix further shows that more false-positive results are obtained by the model, this could be a consequence of taking a maximum length smaller than the number of words in the sentence. Another limitation of the model is that common words are not removed from the dataset before training and predicting. These words do not hold much of the semantics of the questions most of the time and, therefore, might lead to mis-predictions on some question pairs.



(a) Training and development curves (b) Confusion matrix

Figure 8: Results of our base architecture

4.2 Attention

Here, the attention distributions obtained from the base model for two representative question-pairs were examined. It is obvious that the key words in the questions "letter" and "sisters" are receiving attention from the question 2 as they are the key words that define the semantics of the question. Additionally, the two attention matrices have strong attention for the same words that are in different positions. Finally, the distribution show that common words can divert attention erroneously and, therefore, might affect prediction.

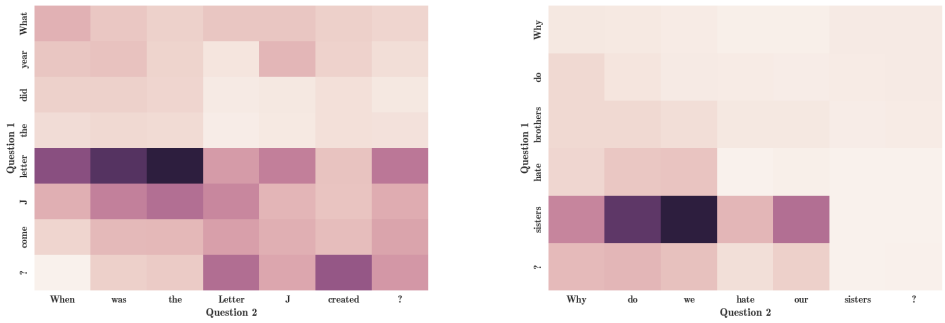


Figure 9: Examples of the attention probabilities

4.3 Hyper-parameter Tuning

Table 3: Hyperparameter tuning results (Red bolded parameters are the parameters different from the base case, Red, bold and underlined numbers represent the best scores, results with * were results where the F1-score was not calculated)

Order	GloVe	Hidden Size	Embed Size	Max Length	Optimize Embedd	Cell	Activation Function	Number Epochs	Dev Accuracy	Dev F1
Normal	6B	50	50	20	Y	LSTM	Tanh	5	0.811	0.752
Normal	6B	**100**	50	20	Y	LSTM	Tanh	5	**0.820**	**0.760**
Normal	6B	**150**	50	20	Y	LSTM	Tanh	10	**0.821**	**0.761**
Normal	**840B**	50	**300**	20	**N**	LSTM	Tanh	6	0.812	*
Normal	6B	50	**100**	20	**N**	LSTM	Tanh	9	0.813	*
Normal	6B	50	**100**	20	Y	LSTM	Tanh	5	0.809	0.747
Normal	6B	50	50	**30**	Y	LSTM	Tanh	5	0.812	0.752
Normal	6B	**100**	50	20	Y	**GRU**	Tanh	5	0.818	0.754
Reverse	6B	**100**	50	20	Y	LSTM	Tanh	5	0.813	0.755
Normal	6B	**100**	50	20	Y	LSTM	**ReLu**	5	0.798	0.740

The results of our hyper-parameter tuning experiments are summarized in Table 3. It was found that the hidden size was the only hyper-parameter that significantly influenced the results positively. Tuning shoed that a hidden size of 100 obtained development accuracy of 0.820 while only a marginal improvement of 0.001 was achieved when the hidden size was increased to 150.

The words in the questions were tokenized using GloVe vectors obtained from [3]. However different version of these GloVe vectors are available. Most of the results were obtained using the smaller vocabulary of 6 billion tokens obtained from Wikipedia and Gigaword 5, while there exists a Common Crawl version which has 840B tokens with an embedding size of 300. It was found that there was a decrease in performance when using this larger corpus. Furthermore, tuning showed that increasing the embedding size did not improve the development accuracy significantly.

Other hyper-parameter tuning such as the RNN cell type (GRU), activation function (ReLu), order of sentence (reverse), did not improve the performance of our model. The results from these experiments proved that the optimal hyper-parameters were found for the chosen final model.

5 Conclusions

- LSTM were used as the building blocks of our architecture.
- GloVe vectors were used as the embedding vectors that were updated during training.
- Three dynamic RNNs were used to capture order dependent relationships of the questions.
- Bilinear global attention was used to focus on important words in the first question [2].
- Hyper-parameters were tuned to obtain best model.
- Development accuracy of 0.82 and F1 score of 0.76 were achieved.
- The final test accuracy was 0.82 with F1 score of 0.76.

6 Future Work

There are several tasks that could help our model improve and are summarized as future work as follows,

- Pre-processing and visualization of the data. Removing the common words from the question pairs before sending them to the constructed neural net would be an immediate choice.
- Construct a conventional machine learning algorithm to predict the labels and compare the results with neural network. Specifically, engineer features such as questions' lengths, KL divergence, distance between questions' vectors etc. and predict using these features.

- Test other neural networks architectures such as convolutional neural networks. This could help because CNN do not care about the ordering of the words and, hence, might give slightly different results.

Acknowledgments

The authors of this project would like to thank Danqi Chen for her valuable comments and discussions. We also would like to thank CS224N team for the teaching and Microsoft Research for providing computational resources.

References

[1] KORNL CSERNAI. First quora dataset release: Question pairs, Jan 2017. URL `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`.

[2] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation.

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