Determining Entailment of Questions in the Quora Dataset

Albert Tung
Department of Computer Science
Stanford University
Stanford, CA 94305
atung3@stanford.edu

Eric Xu
Department of Computer Science
Stanford University
Stanford, CA 94305
ericxu0@stanford.edu

Abstract

Automating the process of finding duplicate questions is one of the most challenging tasks in Natural Language Processing for knowledge-sharing platforms like Quora. An accurate predictor would better organize the forums and make searching and answering questions more efficient. In this paper, we explore the effectiveness of several models from Stanford Natural Language Inference publications on a Quora dataset. We experiment with two main ideas: word ordering and word alignment. The first is tested through a long-short-term-memory (LSTM) recurrent neural network, and the second is implemented with a decomposable attention model. We eventually achieve the highest accuracies when we combine the two designs into one, producing the LSTM attention models.

1 Introduction

Determining if questions imply the same answer or not can improve the ability of machines to understand and reason and also enable knowledge-seekers on forums or question and answer platforms to more efficiently learn and read. Furthermore, answerers would no longer have to constantly provide the same response multiple times.

Finding an accurate model that can determine if two questions from the Quora dataset are semantically similar will be a challenging task given that even humans have difficulty accurately predicting if two questions have the same meaning. In addition, evaluating the data set, we often find that questions have ambiguous meanings and enigmatic symbols when question writers ask about either highly technical subjects or extremely general thoughts. Lastly, we find that many of the questions are not always grammatically correct and have words that are frequently misspelled.

The plan to search for an accurate model begins with using the most popular models from the Stanford Natural Language Inference (SNLI) publications under the section "Three-way classification" https://nlp.stanford.edu/projects/snli/.

2 Background/Related Work

Currently, the approaches for determining sentence entailment on the SNLI dataset have varied from simple feature-based models to complex neural networks that allow for a better extraction of words and sentences. In the previous work of Bowman et al. 2015 [1] LSTM encoder, they achieved a test accuracy of 77.6% which did not beat the unigram and bigram features, but allowed for the introduction of neural network based models that encoded the words in the sentence. By encoding the two sentences, concatenating them, and then using them in a multi-layer neural network for classification, they created a general framework that many papers followed with some modifications.

With the introduction of attention modeling, many research papers such as Rocktschel et al. 2015 [2] better generated alignments between words of both sentences or entire sentences which subsequent models such as Parikh et al. 2016 [3] modified by performing intra-sentence attention which achieved a 86.8 % accuracy.

3 Approach

We implemented several different models including baselines and from various papers that were produced to determine sentence entailment on the SNLI dataset. We used the 840B common crawl GloVe pretrained embeddings `https://nlp.stanford.edu/projects/glove/`, the starter code from CS224N `http://web.stanford.edu/class/cs224n/`, and tuned the hyper-parameters on these various models to achieve the optimal accuracy.

3.1 Bag of Words Model

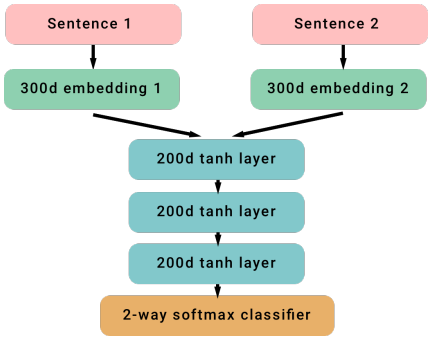

Figure 1: Bag of Words Model neural network representation

Our baseline constructed a dense vector representation for the questions which was then fed into a feed forward neural network to make a prediction. For each question, we summed the 300-dimensional GloVe word embeddings for each word in the question to get a question embedding. Then we masked the question embeddings, concatenated the vectors of the two questions and used the final vector as the input for a neural network with three tanh layers with dropout and a final softmax activation function.

3.2 RNN with GRU and LSTM cell

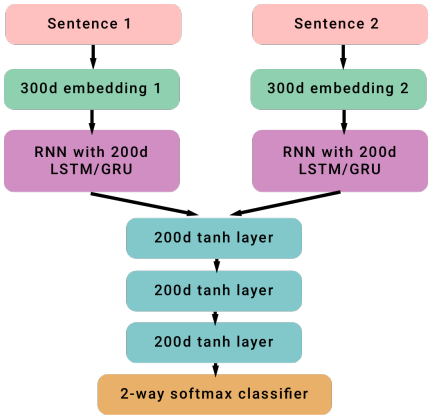

Figure 2: RNN Model with LSTM/GRU

Recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory units (LSTMs) have been used extensively in neural networks such as in Bowman et. al 2015 [1]. Because of their ability to utilize temporal information such as the order of the words in a sentence, our next two approaches used an LSTM/GRU to construct the question representation vector rather than using the sum of the word vectors.

The GloVe embeddings were fed into the LSTM as inputs and we took the final output of the recurrent neural net as the question representation vector. Then similar to the bag of words model, we sent the question vector through a three layer neural net (with tanh as the activation function, dropout, softmax to normalize final prediction). We also experimented with a GRU cell in place of the LSTM.

3.3 LSTM with Attention

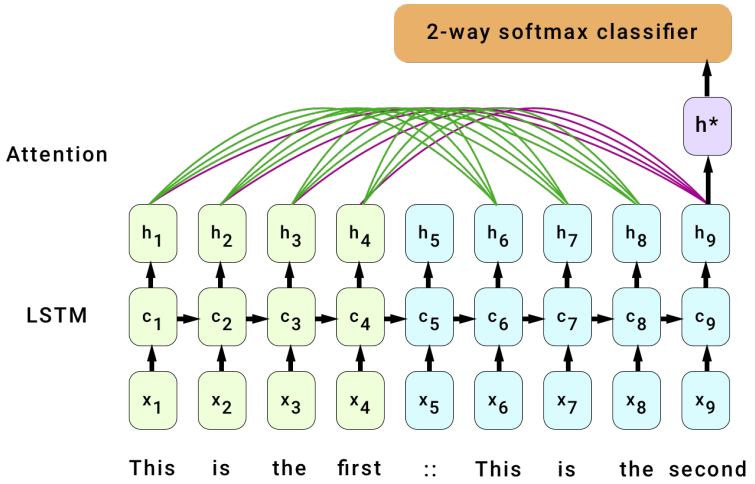

Figure 3: Attention model with two LSTMs (shown in cells green and blue respectively). The purple lines indicate attention while green and purple lines combined demonstrate word-by-word attention. Two-way word-by-word attention is the same as word-by-word attention but with the premise and hypothesis switched and the final h* state concatenated

While LSTMs can store information about the words we have seen in the question, its state cannot capture the complete semantics. As a result, our next approach incorporated LSTM with attention, word-by-word attention, and two way word-by-word attention modeling developed by Rocktschel et al. '15 [2].

The first attention model created two LSTMs. The first question was fed into the first LSTM, and its final hidden state was used as the first hidden state in the second LSTM. The second question was then fed into the second LSTM and let h_N be the final output vector of the second LSTM. Let $Y = [h_1, h_2, ..., h_L]$ where h_i is the output produced by the first LSTM after the ith word is read (L is the length of the first question). We then created weights W^y, W^h, and w to compute the attention matrix α:

$$M = \tanh(W^y Y + W^h h_N \otimes e_L)$$

$$\alpha = \text{softmax}(w^T M)$$

where e_L is an L-dimensional vector of ones and $\otimes$ is the outer product. To obtain the final vector, h^*, that encapsulates both questions, we used two weights W^p and W^x to get

$$r = Y\alpha^T$$

$$h^* = \tanh(W^p r + W^x h_N)$$

Finally, we projected h^* into a two-dimensional vector and applied a softmax layer to get the final prediction.

For word-by-word attention, we created alignment vectors α_t for every word in the second question. In particular, the ith entry of α_t contained the alignment weight between the ith word in the first question and the tth word in the second question. To compute α_t, we created weights W^t for each time step in the second LSTM and looped through every output h_t for the second question:

$$M_t = \tanh(W^y Y + (W^h h_t + W^r r_{t-1}) \otimes e_L)$$

$$\alpha_t = \text{softmax}(w^T M_t)$$

$$r_t = Y\alpha_t^T + \tanh(W^t r_{t-1})$$

To get the final question-pair representation, we took r_L (the last attention-weighted representation of the first sentence) and h_N (the last output vector of the second LSTM) to get

$$h^* = \tanh(W^p r_L + W^x h_N)$$

Again, we projected h^* into a two-dimensional vector and applied softmax to get the final prediction.

The third attention model introduces two-way attention by additionally swapping the two sentences and running the word-by-word attention again. This gave us two final question vectors, h_1^* and h_2^*, which we concatenated and projected into a two-dimensional vector.

3.4 Decomposable Attention Model

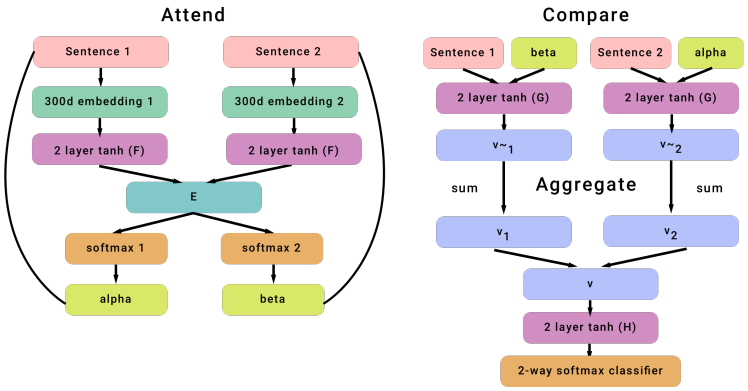

Figure 4: Decomposable attention model representation

Our final approach used the decomposable attention model developed by Parikh et al. '16. With only feed forward neural nets and attention, this model is much simpler and computationally more efficient than the complex LSTM models above and yet performs better on the SNLI dataset.

Let A be the word embedding matrix of the first sentence (rows are the embedding vectors of the words), and let B be the word embedding matrix of the second sentence. For each word embedding vector in the two sentences, we ran it through a feed forward neural net F. This gave us two new matrices $F(A)$ and $F(B)$, and we get E, the unnormalized attention weight matrix, by performing $F(A)F(B)^T$. To generate the soft-alignment between words, we do $\beta = \text{softmax}(E)B$ and $\alpha = \text{softmax}(E^T)A$, where the softmax function is applied for each row in the matrix. As a result, the ith row in β aligns with the ith word in the first sentence, and the ith row in α aligns with the ith word in the second sentence.

We next compared the both of the two soft-aligned matrices by concatenating A with β and B with α. Then we ran each row into another feed forward neural net G to get $\tilde{v}_1 = G([A, \beta])$ and $\tilde{v}_2 = G([B, \alpha])$.

Our last step aggregated the results by creating v_1 and v_2 which are the resulting vectors when the rows of $\tilde{v}_1$ and $\tilde{v}_2$ are summed, respectively. To generate the final prediction, we concatenated v_1 and v_2 to get $v = [v_1, v_2]$, applied a feed forward neural net H on v to get $H(v)$, and then used softmax on the resulting vector.

All three neural nets (F, G, and H) had two tanh layers with dropout.

4 Experiments

4.1 Data

Our dataset was taken from Kornel Csernai's post, First Quora Dataset Release: Question Pairs, at `https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`. The direct URL to the dataset is `http://qim.ec.quoracdn.net/quora_duplicate_questions.tsv`.

Each line in the dataset is in the format

[ID] [Question 1 ID] [Question 2 ID] [Question 1] [Question 2] [is_duplicate]

where "is_duplicate" is either 0 or 1 indicating whether the two questions are duplicates. There are 404290 questions in all, so we used 70% of the questions as the training set, 10% of the questions as the dev set, and the remaining 20% of the questions as the test set.

Each question was split into a list of string tokens using the Natural Language Toolkit `http://www.nltk.org/`. After parsing all the questions, we found that the longest question had length 269 but only 38 questions had length greater than 100 tokens. As a result, we decided to set the max length of all questions to be 100 in the models by ignoring the words appearing after the first 100 tokens. However, for the three attention models presented in Section 3.3, we set the max length to be 50 to speed up computation. In addition, with this tokenizer, we were able to match 72.7% of the words in the Quora dataset with word vectors.

4.2 Training

During training, all weight matrices were initialized with the xavier initializer and the biases were initialized to the zero vector. Initially, we set out-of-vocabulary word embeddings to be values randomly sampled from (-0.05, 0.05) according to a normal distribution. We experimented with fixing the embeddings and fine-tuning the embeddings.

For all of the models except for decomposable attention, we trained with cross-entropy loss with L2-regularization on the weights for ten epochs with a batch size of 100. The decomposable attention model was trained on cross-entropy loss through twenty epochs with a batch size of 50. We used the Adam optimizer with rates $\beta_1 = 0.99$ and $\beta_2 = 0.999$. We tested various dropout rates from 0.1 to 0.8 with a step of 0.1 and L2-regularization betas of 0.001 and 0.01.

Accuracy was measured as number of labels predicted correctly divided by the total number of labels predicted.

4.3 Challenges

We had numerous challenges building the framework and implementing the corresponding models which we wanted to document briefly to show our thoroughness in exploration. One of the major challenges with these models was avoiding memory issues given the size of our dataset, we had to batch train and test as well as reduce the size of our sentences (we initially started with the maximum size, but reduced it to speed up training). Furthermore, we were attempting to address issues with overfitting our model, so we introduced dropout and regularization and tried various values for those. However, we felt that this may have been caused by training on our embeddings, so we also experimented with training on our embeddings vs. not training on our embeddings as well as comparing different GloVe pretrained embeddings. Later on, when performing error analysis, we found that we were not parsing conjunctions correctly (i.e. "What's" became "Whats"), so we ended up using a standard tokenizer from NLTK. Lastly, we realized the importance of choosing the right type of activation functions for models such as the decomposable attention model or else our model would refuse to train because of extremely large gradients.

4.4 Results

Table 1: Results on the Quora dataset using various approaches where k is the dimension of the hidden states

Model	k	Train	Dev	Test	F1 Score
Bag of Words	200	96%	80.5%	80.6%	0.8488
LSTM	200	81%	78.6%	78.4%	0.8339
GRU	200	81%	78.4%	78.4%	0.8360
LSTM with Attention	100	95%	81.4%	81.0%	0.8516
LSTM with Word by Word Attention	100	89%	81.3%	81.2%	0.8550
LSTM with Two Way Word by Word Attention	100	94%	81.7%	81.4%	0.8523
Decomposable Attention Model	200	87%	80.0%	79.8%	0.8365

The results shown are the highest accuracies achieved on the development set while training and its corresponding training accuracy. The test set was evaluated based on the restored weights from the best development set result. We also want to note that the truth labels may not always be correct, thus introducing noise.

Parameters: For all of the models, the best accuracies were achieved when we fix the learning rate to be 0.001 and train on the word embeddings. For the bag of words model, we had 0.001 and 0.2 for the L2-regularization beta and the dropout keep probability respectively. For the LSTM/GRU recurrent neural network, we had $\beta = 0.01$ and dropout keep probability of 0.5. For the three LSTM models with attention, we had $\beta = 0.01$ and dropout keep probability of 0.3. Lastly, for the decomposable attention model, we had a dropout keep probability of 0.8.

4.5 Sentence Analysis

We noticed that all of our models were having issues classifying questions such as "Is coding-parks.com legit" and "Is theidealmeal.com legit" which are similar phrases grammatically but with one different keyword. We believe that this is caused by the fact that these words are out of the vocabulary, and their randomly generated embeddings may have been similar. However, we did notice that the word-by-word and two-way-word-by-word attention models made fewer mistakes on these types of sentences.

On the other hand, our LSTM attention models improve on aligning words that are tangentially related, so the sentences "What is EBD for vehicle" and "What is EBD in cars" is classified correctly as the same question unlike all of our other models.

But when the order of the words is different, the decomposable attention model fares better. For the question pair "How is life in prison" and "What is prison life like", the decomposable attention model is able to align the corresponding words across the questions and predict they are the same question. However, the LSTM models are unable to decode the different sentence structure between the two questions.

4.6 Attention Visualizations

For the decomposable attention model, we are able to see how the model align words by extracting the attention weights of the matrix E after the softmax function is applied across its rows.

The top left plot in Figure 5 shows that the model not only successfully aligns the same word together (meditation), but also synonyms such as "is/does" and "help/useful". In addition, it is able to group the phrase "how does" with "is".

Phrase grouping is even more prominent in the bottom two plots. In the lower left plot, "is a" corresponds to "is" and "finance rate" aligns with "finance". Likewise, it infers that "spillatore" is related to beer in the lower right plot.

Finally, we see that in the upper right plot, the model can even align similar words that don't appear in the same relative positions in their respective questions. In particular, it matches "am" with "is" and they are both the main verbs in the questions.

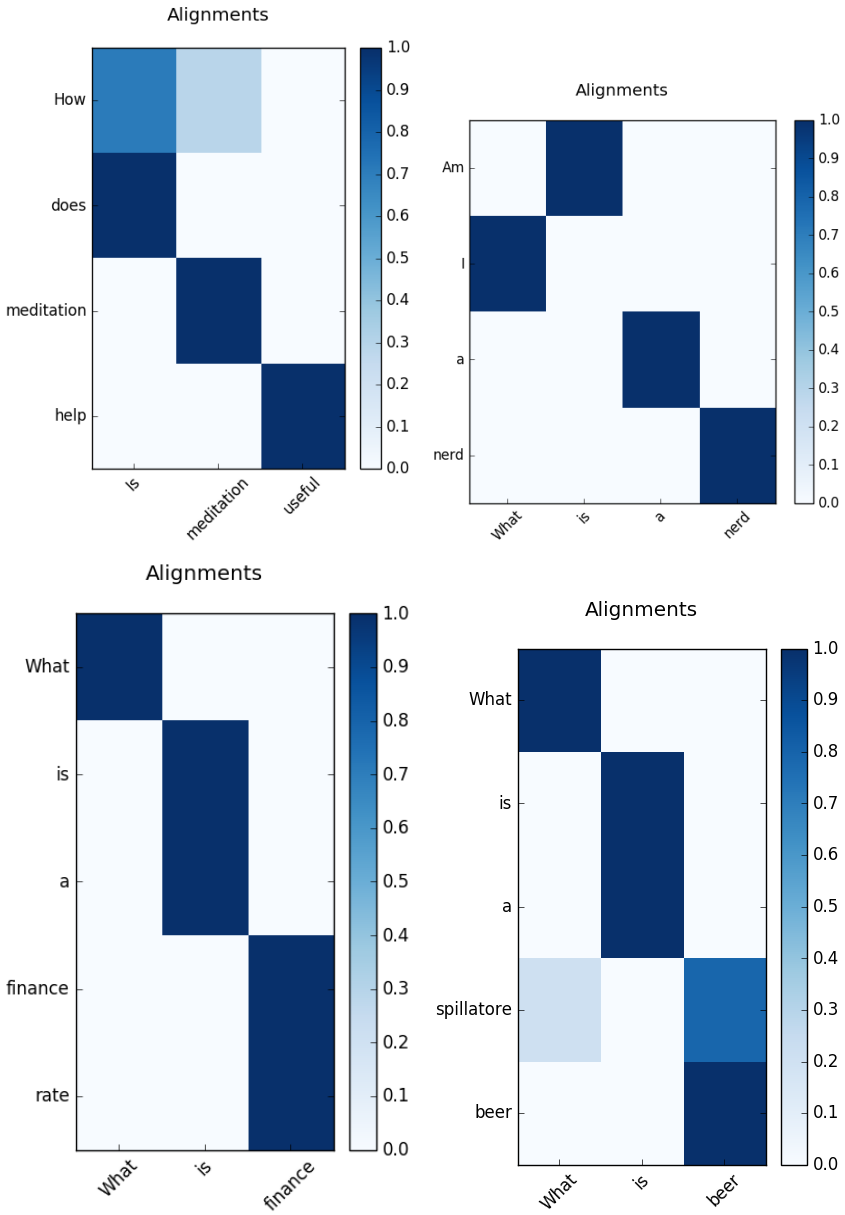

Figure 5: Soft-alignment Visualizations

5 Conclusion

Surprisingly, the simple bag of words model that sums the word embeddings does really well on the Quora dataset. But excluding the baseline, there are two main ideas that used here for predicting whether two questions are semantically equivalent.

The first is storing and utilizing temporal information - in particular, the order of the words in the question. This leads to the GRU/LSTM model, which predicts around 78.4% of the questions

correctly. However, the LSTM does not relate the words across the two questions, which is the second big idea for classifying the question pairs.

Word alignment and attention is the focus of the decomposable attention model, and it outputs a prediction based on how well it can match up similar words across the questions. However, it disregards the ordering of the words, and thus, produces a small improvement up to 79.8% accuracy.

As a result, by combining the two ideas, we are able to achieve a higher accuracy, as shown through the LSTM with Attention, LSTM with Word by Word Attention, and LSTM with Two Way Word by Word Attention. The LSTM with Attention only attends the second sentence as a whole with respect to each word in the first sentence. Thus, by applying word by word attention, we are able to increase the accuracy a bit more. Finally, by swapping the questions and generating another set of alignment weights, we can do even better.

Nevertheless, other models may prove to attain higher test accuracies on the dataset. Our next steps include implementing intra-attention to the decomposable attention model so that we could use temporal information and testing out a bidirectional LSTM so that we can scan the questions in both directions. We also want to explore ways to reduce mistakes on obscure vocabulary words.

Acknowledgments

We would like to thank our project mentor Danqi Chen for providing us with valuable project advice and giving us suggestions for moving forward with the various models. Lastly, we would like to thank the CS 224N staff and instructors for helping make this class and project possible. We have learned a great deal from understanding papers and learning new models.

References

[1] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015.

[2] Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, and Phil Blunsom. Reasoning about Entailment with Neural Attention. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2015.

[3] Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016.