Too Many Questions

Ann He
Undergraduate
Stanford University
annhe@stanford.edu

Jeffrey Zhang
Undergraduate
Stanford University
jz5003@stanford.edu

Abstract

Much work has been done in recognizing the semantics of sentences as well as semantic relationships between sentences. We applied some of these previous approaches to the space of question similarity. Our task was to create a classifier that given a pair of questions, attempted to predict whether or not the questions were asking the same question. Of the four models we attempted (bag of words, LSTM with distance and angle, LSTM with normal attention, and LSTM with word-by-word attention), the model that performed the best was LSTM with distance and angle, achieving a test accuracy of 84.7 percent. Word-by-word attention also outperformed normal attention by almost 1 percent.

1 Introduction

Detecting duplicate questions have many applications in industry, especially for question-answering services. Quora, one such question-answering application, receives several thousand questions daily, a significant portion of which are questions that have been asked before. Thus, to encourage a better model to detect these duplicate questions, they released a dataset of over 400,000 labeled question pairs. A question pair is labeled with 1 if the two questions are asking the same question and 0 if not. Our task is to train a classifier which, given a question pair, predicts whether or not the two questions are the same with as high probability as possible.

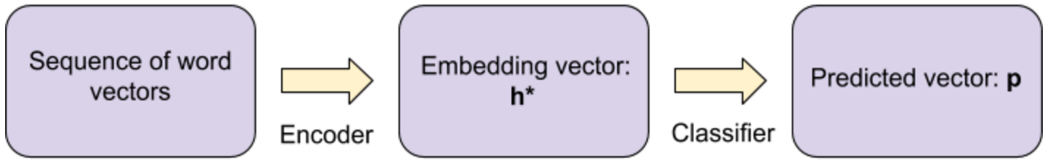
1.1 Background/Related Work

The problem of semantic similarity of question pairs falls under the umbrella of natural language inference, a central area of artificial intelligence. Work in this area focuses on informal reasoning, lexical semantic knowledge, and accounting for variation in natural language expressions. The task of recognizing textual entailment (RTE) is to determine whether two sentences (called the premise and the hypothesis) are related, contradictory, or not related at all. RTE is motivated by important problems in information extraction, text summarization, and machine translation. Our project uses techniques inspired by the paper Reasoning about Entailment with Neural Attention by Rocktaschel, et al. Before the work of these researchers, natural language systems for RTE relied on heavily engineered pipelines and features–Rocktaschel et al are one of the first to build an end-to-end differentiable model for RTE and assess it on high-quality datasets. In particular, they used long short-term memory units with attention mechanism.

Long short term memory cells are a recent excitement in artificial intelligence, and are particularly applicable to problems where the input is sequential–such as natural language. They are specialized neural network with additional gates which to allow for the removal or addition of information to the current cell states. Attention mechanism provides an even more sophisticated method of contextualizing by assigning weights to the importance of each word in a sentence given its paired sentence. Word by word attention allows for more fine grained dependencies, processing each word in the current sentence with attention weights over words in the pair sentence. We implement both normal attention and word-by-word attention with our LSTM models.

2 Approaches

All of our approaches have the same general model. We start by tokenizing the questions and using GloVe 6b.50 word embeddings. The variety comes in how we generate an overall embedding for the two questions (h^*). We then feed h^* into some standard classifier to get a predicted vector p. Below is a visualization of the general workflow for each of our approaches that differ in how encoding and classification is performed. For all of these approaches except bag of words, the loss is defined as cross entropy loss plus an L2 regularization term.



2.1 Bag of words

The bag of words approach serves as the baseline for our project. For each question pair, we take the sum of the word vector embeddings and concatenate the two resulting vectors together to get h^*. More formally, given word vectors of the first question $u_1, u_2, \cdots, u_{L_1}$ and word vectors of the second question $v_1, v_2, \cdots, v_{L_2}$, we define h^* as:

$$h_1 = \sum_{i=1}^{L_1} u_i$$
$$h_2 = \sum_{i=1}^{L_2} v_i$$
$$h^* = h_1 \parallel h_2,$$

where $\parallel$ denotes concatenation.

Finally we used the built-in sci-kit learn random forest classifier on h^*.

2.2 LSTM encoder

Simply summing word vectors together is a rather naive approach as it is incapable of capturing temporal and other complex aspects of the sentence. It is shown in (Pagliardini et. al.) that an embedding produced by an LSTM encoder actually ends up producing a weighted average of the word vectors in a sentence.

Our first approach involved passing the first question through one LSTM and the second question through another and taking the final states of both: h_1, h_2. Then we simply concatenated the two vectors (i.e. $h^* = h_1 \parallel h_2$) and passed h^* through a tanh layer followed with a softmax classifier.

We found that concatenation doesn't really capture the relationship between the two vectors very well, so we tried using the approach described in (Dandekar et. al.) by taking the concatenation "distance" and the "angle" between the two vectors as h^*. More formally, we let:

$$h^* = |h_1 - h_2| \parallel (h_1 \circ h_2),$$

where $\circ$ denotes elementwise multiplication and $|h_1 - h_2|$ is the absolute difference between h_1 and h_2.

We then pass h^* through a tanh layer followed with a softmax classifier to get a 2-dimensional prediction vector p:

$$p = \text{softmax}(W_f \tanh(W_h h^* + b_h) + b_f).$$

2.3 LSTM encoder with attention

Here we used the normal attention model described in (Recognizing textual entailment). We use the first question as the "premise" and the second question as the "hypothesis".

As described in the paper, we first obtain a matrix M which is a nonlinear combination of the outputs of the LSTM over the first sentence with the final state of the LSTM over the second sentence:

$$M = \tanh(W_y Y + W_h H),$$

where Y is a matrix of L columns for the outputs produced by the LSTM while processing the first question and H is a matrix of L columns which are all copies of the final state of the LSTM after processing both questions.

We also learn an attention vector α defined as:

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

where w is a learned vector.

Next, we use the attention weights to get a representation r, which is simply the corresponding linear combination of the outputs of the LSTM on the first question.

$$r = Y\alpha^\top$$

Finally, we obtain h^* in much the same way as we obtained M:

$$h^* = \tanh(W_p r + W_x h_N),$$

where h_N is the final state produced by the LSTM after processing both questions.

After obtaining h^* we passed it directly into a softmax classifier.

2.4 LSTM encoder with word-by-word attention

Normal attention doesn't capture word to word weights between the two questions, so we decided to try word-by-word attention as described in (Recognizing textual entailment).

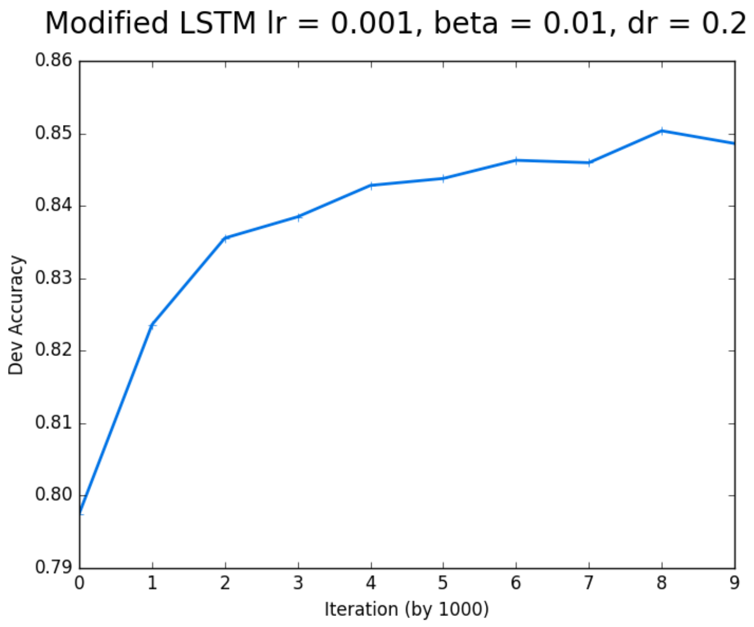
This approach was largely similar to normal attention except applied at each time step corresponding to each word in the second sentence. The calculation of the representation vector r^t also uses r^{t-1} by adding it in the calculation of M_t as well as explicitly in the calculation of r^t.

Again, we pass in our final embedding h^* into a softmax classifier.

3 Experiments

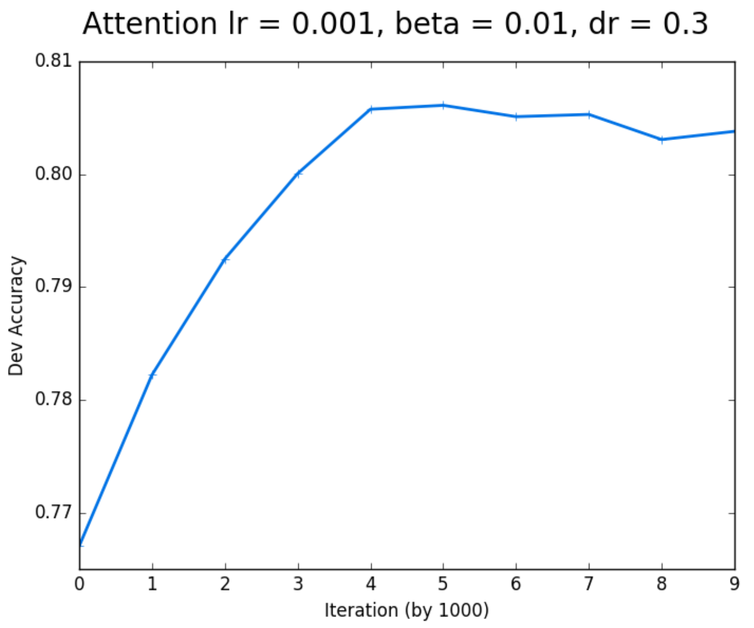
3.1 LSTM

The original approach for the LSTM which concatenated the hidden representations of the two sentences achieved test accuracy 77.38%. Modifying to Dandekar et. al.'s approach, which takes the concatenation distance measures of the hidden vectors with the angle, we were able to achieve 84.71%. The plot of the dev accuracies for the second model during training are shown below with a table detailing the values of the max train accuracy and the dev score for each epoch (3.4).



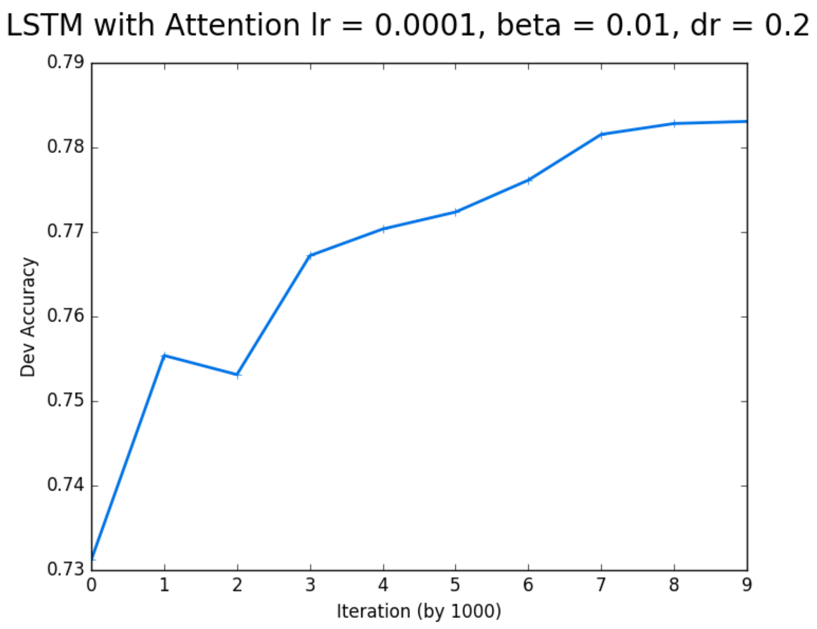
3.2 LSTM with Attention

Our original LSTM with attention model achieved 77.66% test accuracy. However, after changing our train:dev:test ratio from 2:1:1 to 70:15:15 we were able to improve test accuracy to 79.85%. Below we plot the dev accuracy during the training of the improved model.



3.3 LSTM with Word-by-Word Attention

We tried three different hyperparameter settings. (put them all in table). Below is a plot of the dev accuracies for each epoch of the first set of hyperparameters.



3.4 Aggregated Results

Country List		
Model	Best Dev Acc	Test Acc
BOW	N/A	0.8036
LSTM	0.7722	0.7738
Modified LSTM	**0.8503**	**0.8471**
Attention	0.8061	0.7985
Word-by-word (lr = 0.0001)	0.7830	0.7835
Word-by-word (lr = 0.0005)	0.8012	0.7982
Word-by-word (lr = 0.001)	0.8059	0.8046

Note there is no dev accuracy for the bag of words model since there are no hyperparameters to tune as we pass the summed vector directly into a the built-in sci-kit learn random forest classifier.

3.5 Analysis

Looking at examples of sentences that were misclassified do not really help us get an idea of why the model underperforms, but it is nevertheless interesting and worth noting. Our word-by-word attention model incorrectly classifies the following question pairs:

```
1. What are some examples of sentences using the word "simile"?
2. What are some examples of sentences using the word "nevertheless"?
```

We predicted that the questions mean the same thing, but the last words mean very different things, so the questions are not the same. Our model is unable to recognize that the difference in the last word simply causes the two questions to mean entirely different things.

```
1. How will it be after death? Where does the soul go?
2. What happens to the soul after it leaves the body?
```

Again we predict that the questions mean different things, but they actually mean the same thing. We believe that the reason we misclassify this is because the first question is phrased as two questions, which makes the alignment very difficult to recognize.

4 Conclusion

As our results show, we were able to beat our bag of words model baseline by a using an LSTM encoder with distance and angle calculated as described in Dandekar et al as well as with a word-by-word attention model. We also saw that the LSTM encoder with distance and angle had a significantly better accuracy than any of our other models, and we attribute that to the fact that most of our questions were short enough for there not to be an improvement with recognizing alignments through attention. However, word-by-word attention still outperforms normal attention as expected, since word-by-word essentially is normal attention applied for each word of the second sentence.

In the future, we plan to explore more approaches, including using tree-structured LSTM's as described in (Tai, Socher, Manning). Also, the dataset has several words misspelled (thus not in our dictionary), which affects our results. We can autocorrect these spellings as a preprocessing step.

In terms of use as an industry application, there could be more sophisticated models that detect not only based on the question statements but also answers posted to determine if questions are semantically the same. This is where alignments and more complex models such as LSTM encoders with attention could outperform a simple LSTM model with distance and angle. The model could then be applied to collapse or merge questions with the duplicate version that has already been asked.

References

[1] Rocktaschel, T. & Grefenstette, E. & Hermann, K.M. & Tomas, K. & Blunsom, P. (2015) *Reasoning about Entailment with Neural Attention.* Web: ArXiv.

[2] Dandekar, Nikhil (2017) *Semantic Question Matching with Deep Learning* Web: Engineering At Quora.

[3] Pagliardini, Gupta, Jaggi (2017) *Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features.* Web: ArXiv.

[4] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. *Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks.* In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing.