Question Encoding for Robust QA Networks

Arnav Vaid[1], Anaga Rajan[2], Christopher Correa[3], and Nikhil Sharma[4]¶

May 20, 2018

1 INTRODUCTION

There are many ways to syntactically ask the same question. Can we train a machine learning model to map similar questions to a single syntax-invariant representation? We propose using neural techniques to learn a mapping from a typical word-embedding question representation to a syntax-invariant one.

Much progress has been made in learning semantically meaningful representations of words in continuous vector spaces. In this spirit, efforts are being made to learn similar representations for sentences. These embeddings can be learning in an unsupervised manner (techniques like SkipThought [3]), while others can be learned through training for a specific task.

We propose to learn these embeddings by training on sentence similarity tasks. Using a Self-Attentive RNN model [1], we construct an output sequence of new embeddings represented as matrix M. We propose an addition to this network which condenses variable length embedding sequences to a single vector representation, which we call the summary vector, allowing explicit comparisons between two questions. We hypothesize that this intermediate embedding sequence M produced by the model will result in a more syntactically invariant representation of a question.

To evaluate the utility of our embeddings, we attempt to use them to train Question-Answering (QA) network. QA networks are tasked with finding the answer to a given question within an article. Good performance on this task requires a good understanding of how syntax affects semantics (Did the cat bite the dog? vs. Did the dog bite the cat?), as well as distilling question meaning in a syntax independent way. We hypothesize that a QA network might converge faster or achieve more robust performance if fed our new embeddings.

We train the Question Encoding network on the Quora Question Pairs dataset, and train the Question Answering network on the Stanford Question Answering Dataset (SQuAD). This should be sufficient data for our purposes; the Quora Question Pairs dataset consists of over 400,000 question pairs and SQuAD is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles containing 10,000+ question-answer pairs on 500+ Wikipedia articles.

**This work was not supported by any organization

†[1]A. Vaid is with the Department of EECS at the UC Berkeley

‡[2]A. Rajan is with the Department of EECS at the UC Berkeley

§[3]C. Correa is with the Department of EECS at the UC Berkeley

¶[4]N. Sharma is with the Department of EECS at the UC Berkeley

2 APPROACH

2.1 Data Preparation

In order to first generate our sentence-similarity embeddings, we used a Bidirectional LSTM with self-attention which is typically used for sentence matching tasks [1]. This model was trained on the Quora Question Pairs dataset which contains a series of question pairs and binary label that indicates whether the questions mean the same thing. Then, in order to train our DCN model, both for the baseline and the experimental models, we used the SQuAD (Stanford Question Answering Dataset) which contains a series of questions and reference documents.

Unfortunately, the Quora Question Pairs dataset didn't overlap with the SQuAD dataset and we were unable to find a questions-answers dataset with similar question labels as well, therefore we couldn't perform end-to-end training with our model. Instead, we trained the sentence-embedding model separately, and then used the model when preprocessing the data which was fed into the training process of the DCN.

Furthermore, we used the 300-dimension, Global Vectors for Words Representation (GloVe) to embed our sentences throughout the project, in both the Sentence-Similarity model and the DCN.

2.2 Baseline Model

Our baseline Dynamic Coattention model is already the current state-of-the-art question answering model [2], shown in Figure 2. The model has co-dependent modules that encode the question and document. Then, the outputs are given to the co-attention module that iteratively makes improvements to a dynamic pointer that narrows down on the relevant parts of the reference text. The paper reports that the model achieves a 75.9 F1 score on the SQuAD dataset. When we train the DCN model, we achieve approximately a 62 F1 score.

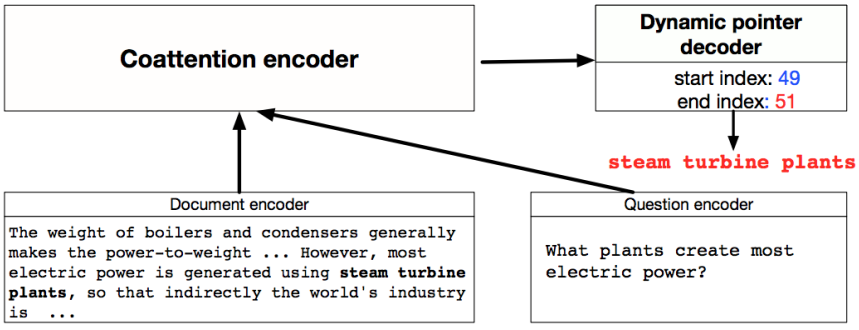


Figure 1: Dynamic Pointer Decoder

2.3 Question Encoding Network

Our QE Network uses an architecture described in [1] consisting of an LSTM that generates a sequence of hidden states, consolidated as a matrix H. Self-attention is computed using a

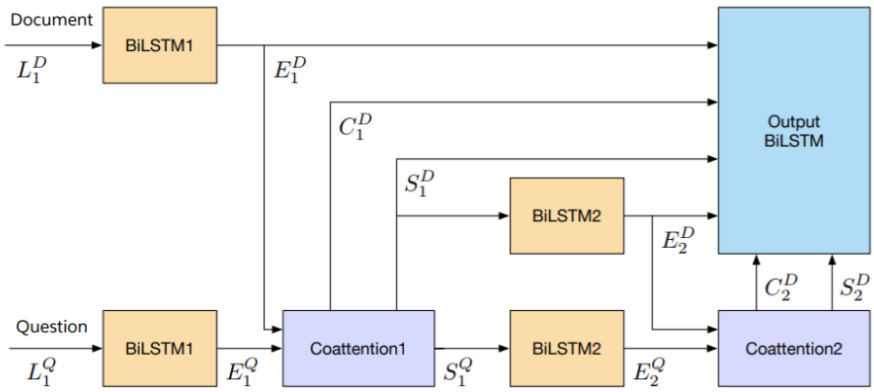


Figure 2: DCN Model Architecture

feed-forward network to produce matrix A, which is then multiplied by H to produce M, the sentence embedding. The architecture is visualized in Figure 3.

$$
\begin{aligned}
H &= (h_1, h_2, \ldots, h_n) \\
A &= softmax(W_{s2} tanh(W_{s1} H^T) \\
M &= AH
\end{aligned}
\tag{1}
$$

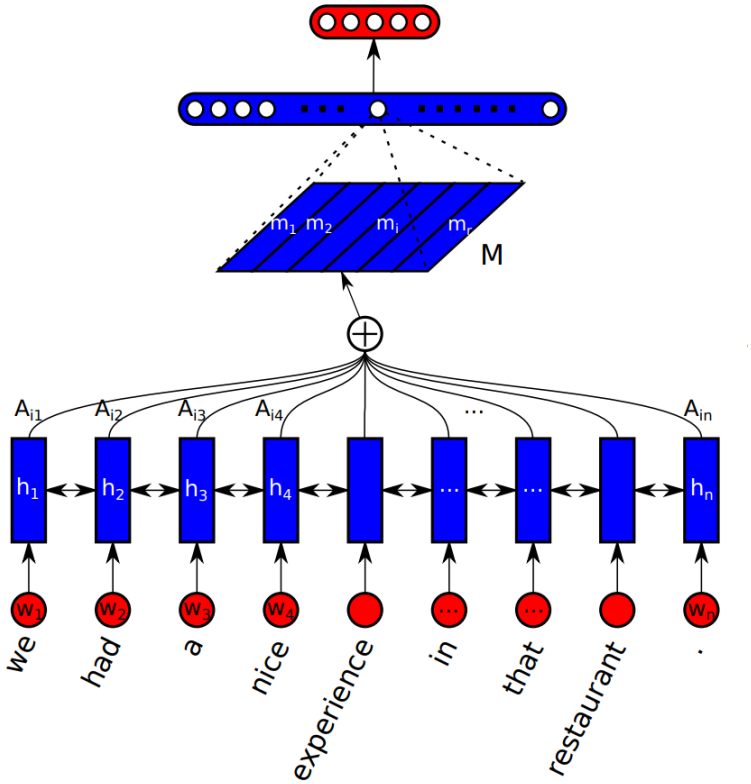


Figure 3: Sentence Embedding Architecture

Given our training data consists of question pairs, our network needs to generate an sequence-length invariant representation of the questions. We propose to do this by computing vanilla

attention over the columns of the sentence embedding M to get the 'summary vector' m as shown in Figure 4. We use an L1 similarity metric as input to a binary cross-entropy loss function to train our network. By training on the task of identifying similar questions, we hypothesize that the M matrix will learn a syntax invariant representation of the question, and m will capture the semantic meaning of a question.

2.4 Final Model

The goal for the final model was to utilize the sentence-similarity representations from the Bi-directional LSTM to create a semantics-free representation that could be used to improve the performance of the DCN. The high-level idea is portrayed in Figure 5. Ideally, the first model would strip away the syntax variations between similar questions and leave behind a representation that only carried the meaning of the question. We have attempted several methods to incorporate this representation in the DCN inputs.

(a) In our initial model, we decided to integrate the sentence-embeddings by first training the Bidirectional LSTM. Then, when training our DCN model, instead of passing in the questions directly, we first run it through our sentence-embedding model and then use the M matrix embedding as our question. This M matrix is a series of weighted sums over the hidden states in the LSTM model. In practice, we believed that the columns of M encoded the meaning of the sentence in a syntax-free manner.

(b) Our belief is that the the DCN was unable to train adequately because the second attention over the columns of M might mean the column orderings would not be prioritized in the Question Encoding Network loss function, but would be necessary for the DCN model. Therefore we decided to input to the DCN the original GloVe embeddings, but append the summary vector m to the end of the sequence. We believed this would get rid of the problem of column ordering affecting the network.

(c) This again didn't produce as well of a result as we wanted, as our performance stayed pretty similar to the baseline. Our next iteration carried two major changes: (1) We reverted back to the M-matrix and appended it to the glove-embedded question and (2) Used two separate LSTMs in the DCN to train the document and question (due to differing input dim sizes).

(d) After this trial, we decided to take a closer look at what our Bidirectional LSTM was attending to, in hopes of figuring out why our DCN wasn't able use the embeddings to decipher the question. We used an attention-visualizer and found that the model was significantly attending to the padding around H, the aggregate of our hidden states. This padding was intended only to bring all queries to the same length and encoded no information. Therefore in our final iteration, we masked the attention over the padding. We additionally reverted back to using our summary vector appended to our question similar to the second model but kept the separated LSTM architecture from the third model.

3 RESULTS

We found that simply training the DCN on just the question embeddings led to the model performing very poorly. We identify two potential reasons for this. We believe the learned question embeddings are well suited for characterizing the sentence into a summery, but individually do

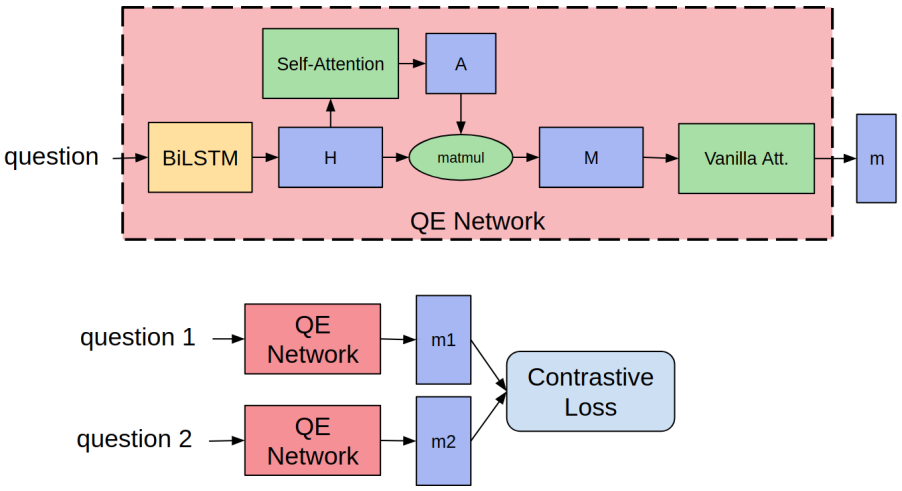


Figure 4: Question Answering Network Architecture

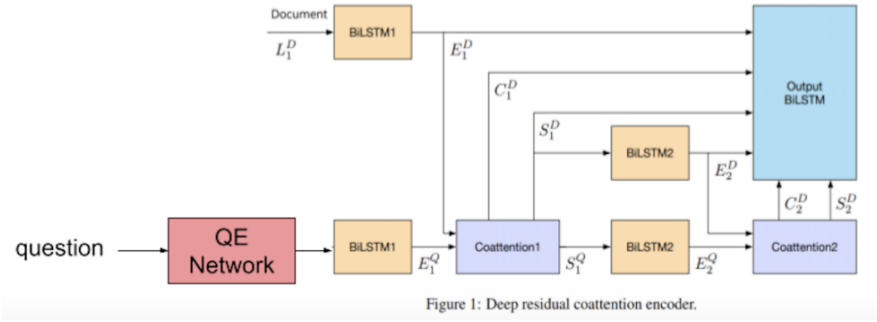


Figure 5: DCN with Question Encoding Network as input

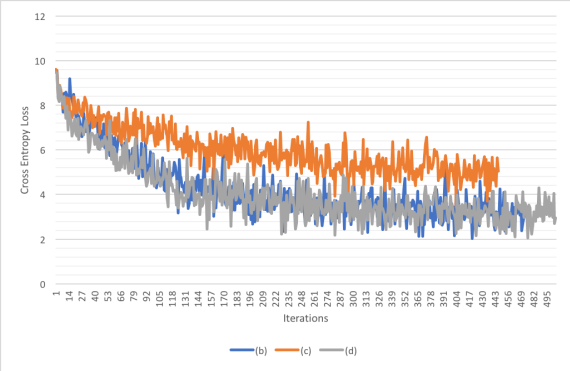


Figure 6: QE Net Cross Entropy Loss

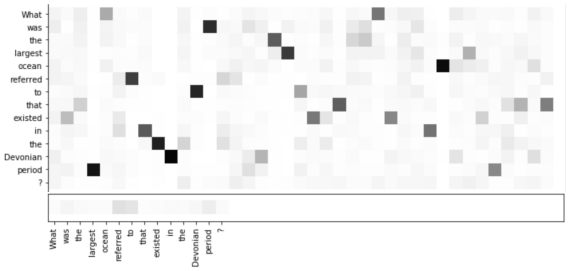
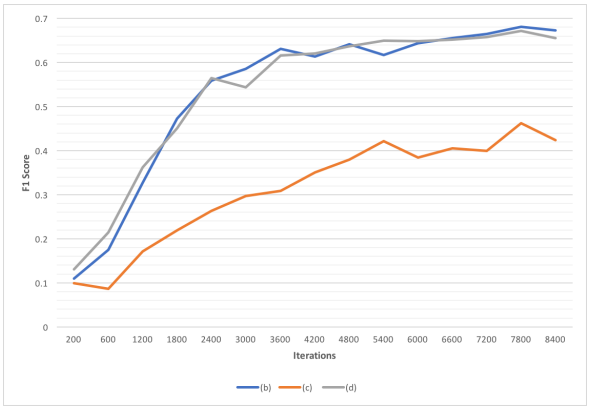
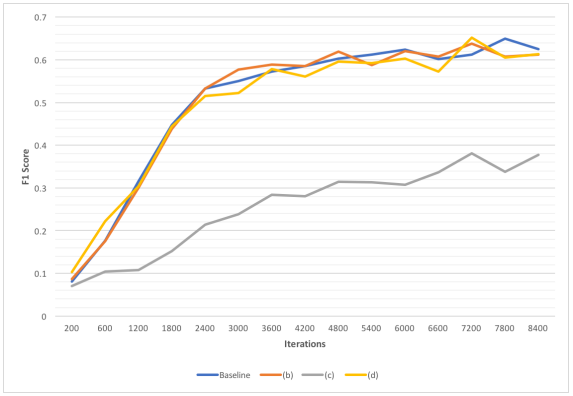


Figure 7: Attention Maps for A Matrix (above) and Vector a(below)

not retain the word relationships that GloVe encodes. Second, we believe of the second attention module, the columns of M of similar questions dont need to be in the same order to be classified as the same, but should be in the same order for the question answering network. The modifications to include the Question Encoded Summary Vector was able to achieve comparable performance to the baseline, but was not able to significantly beat the baseline. Modifications such as the separate LSTMs for the Question and Document and masking the Attention were



(a) Training F1 Score



(b) Validation F1 Score

Figure 8: F1 Scores for Models in Section 2.4

not able to help increase performance.

4 TOOLS

This project borrowed from Github repos containing existing frameworks for processing the Quora Question Pairs Dataset, Siamese BiLSTM architecture for the Question Encoding Network, as well as the DCN for our baseline QA Network. Tensorflow was used due to its mature library of functions and because our group was most familiar with it. Pandas, NLTK, and Matplotlib were used for preprocessing and visualization. We used the AWS AMI instance for quite training of our networks.

5 LESSONS LEARNED

The biggest hindrance to gaining working results was the time taken to train the models. The Question Encoding Network and DCN took ∼4 hours and ∼8 hours respectively to reach saturation. As a consequence the design iteration time was slow, and hyperparameter searches were difficult to do.

Additionally, while borrowing implementations off Github eased the initial burden of writing preprocessing and training pipelines from scratch, it was also a pain to integrate repos, especially when it came to importing models. As a result we resorted to messy hacks that were not the best way to design the project.

Both models also used different data sources with different vocabularies which was hard to coalesce together, for we didn't have any end-to-end datasets which could be used to test our entire model and validate efficiently.

Finally, we learned that attention on variable length sequences mixed with contrastive-style loss can be tricky since attention weights can be learned that correspond to meaningless padding vectors. Future avenues of research would be to utilize transformer networks, which contain multi-headed attention modules that could be more expressive that our model.

5.1 Future Work

It is our hope that our ability to match the baseline results shows significant promise, and that additional hyperparameter tuning and minor modeling assumptions under our defined syntax-invariant paradigm will allow our model to exceed the current state-of-the-art result for Question Answering.

6 TEAM CONTRIBUTIONS

1. **Christopher Correa**: Built QE Network, Integrated and Modified DCN, Wrote Report, Handled AWS training. 40%
2. **Arnav Vaid**: Built QE Network, Integrated and Modified DCN, Wrote Visualizations, Wrote Report. 40%
3. **Anaga Rajan**: AWS training, contrastive loss, Wrote Report. 15%
4. **Nikhil Sharma**: Wrote Report. 5%

References

[1] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130, 2017

[2] Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604.

[3] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In NIPS, 2015.