Natural Language Question-Answering using Deep Learning

Bowen Liu
Department of Chemistry
Stanford University
Stanford, California, CA 94305
liubowen@stanford.edu

Fengjiao Lyu
Department of Mechanical Engineering
Stanford University
Stanford, California, CA 94305
fengjiao@stanford.edu

Rajarshi Roy
Department of Electrical Engineering
Stanford University
Stanford, California, CA 94305
rroy@stanford.edu

Abstract

There has been significant recent progress on applying end-to-end neural network based models for solving question answering tasks. We propose a model that consists of a coattention encoder which captures the interactions between the question and the context, and we introduce a novel multilayer feed forward neural network decoder that estimates the answer span in a single pass. On the SQuAD test dataset, our model achieves a single model performance of 52.8% EM and 64.5% F1.

1 Introduction

Question answering (QA) is an important task in natural language processing where the aim is to build computer systems that can automatically answer questions that are posed in a natural language. It requires both natural language understanding and contextual world understanding. The automatic comprehension of text allows insights to be extracted from raw text data and thus has many real-world applications.

Previous publicly available QA datasets were human annotated and relatively small in size, which made them unsuitable for high capacity models such as deep neural networks. Recently, researchers released the Stanford Question Answering dataset (SQuAD), a reading comprehension dataset that consists of questions posed by crowd workers on a set of Wikipedia articles (Rajpurkar, Zhang, Lopyrev, & Liang, 2016). This dataset consists of 107,785 question answer pairs on 536 articles, which is significantly larger than all the previous human annotated datasets. The size of the SQuAD dataset has enabled the development of much more expressive models for the QA task.

A unique feature of SQuAD is that all answers are entailed by the corresponding contexts. Also, compared with other question answer datasets, where the answers are single words or entities, SQuAD answers can be much longer phrases and often include non-entities. The QA task with the SQuAD dataset can be formulated as identifying the span of words in a document (context) that answers a given question.

This work introduces a novel end-to-end neural network for the QA task. The question and context are encoded using recurrent neural networks and then combined to form a representation that captures the interactions between the question and context. This is heavily inspired by the recently published Dynamic Coattention Networks (DCN) model (Xiong, Zhong, & Socher, 2016). We introduce a novel multilayer feed forward neural network decoder that subsequently calculates the probabilities of all the possible answer start and end index pairs in the context, and then picks the pair with the highest probability as the final answer span.

2 Our Model

We first describe the encoders for the question and context, and subsequently the coattention mechanism and the feedforward neural network decoder that generates the answer span.

2.1 Document and Question Encoder

Similar to (Xiong et al., 2016) the question and context are represented as a sequence of word vectors, $\left(x_1^Q, x_2^Q, \cdots, x_n^Q\right)$ and $\left(x_1^D, x_2^D, \cdots, x_m^D\right)$ respectively. The question was encoded using a bidirectional LSTM, $q_t = Bi-LSTM_{enc}\left(q_{t-1}, x_t^Q\right)$. The intermediate question encoding matrix is defined as $Q' = \left[q_1, q_2, \cdots, q_n\right] \in \mathbb{R}^{\ell \times n}$. A non-linear projection layer was then applied to Q\` to result in the final question encoding matrix, $Q = \tanh\left(W^{(Q)} Q' + b^{(Q)}\right) \in \mathbb{R}^{\ell \times n}$, where l is twice the size of the hidden unit of the corresponding unidirectional LSTMs in the bidirectional LSTM encoder. This layer allows for variation between the question encoding space and the document encoding space.

In order to share representation power, the same bidirectional LSTM was used to encode the context as $d_t = Bi-LSTM_{enc}\left(d_{t-1}, x_t^D\right)$. The resulting context encoding matrix is defined as $D = \left[d_1, d_2, \cdots, d_m\right] \in \mathbb{R}^{\ell \times m}$

Unlike the reference DCN model, we do not include the sentinel vectors in Q, Q\` and D. Additionally we utilize a bidirectional LSTM instead of a unidirectional LSTM to encode the question and context.

2.2 Coattention Encoder

The coattention mechanism is adapted from (Xiong et al., 2016). It simultaneously attends to the question and context, and then fuses both attention contexts. The coattention encoder is illustrated in Figure 1.

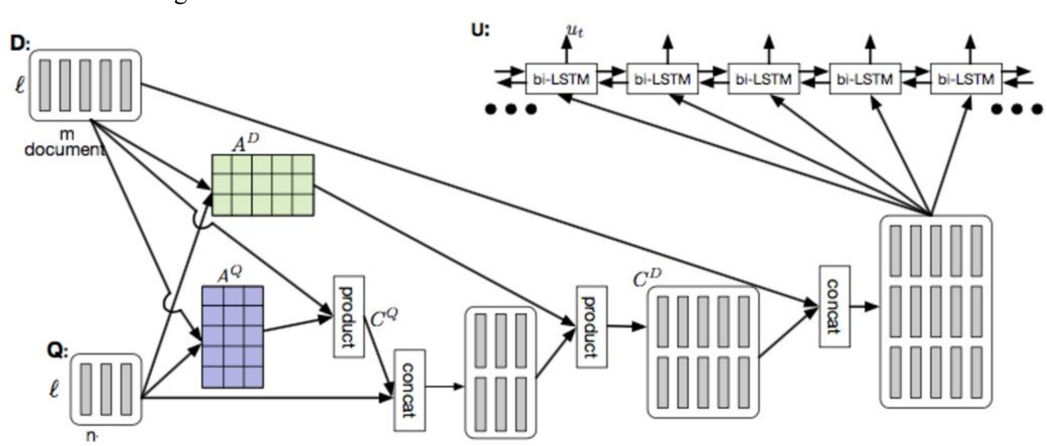

Figure 1: Coattention encoder adapted from (Xiong et al., 2016). The normalized attention weights A^D and A^Q are shown directly while the affinity matrix L is not shown

The question encoding matrix Q and the context encoding matrix D are used to compute the affinity matrix $L = D^T Q \in \mathbb{R}^{m \times n}$, which contains affinity scores that correspond to every pair of document words and question words.

Afterwards, the affinity matrix is normalized column-wise to result in the attention weights A^D across the question for each word in the context, and normalized row-wise to result in the attention weights A^Q across the context for each word in the question.

$A^Q = softmax(L) \in \mathbb{R}^{m \times n}$ and $A^D = softmax(L^T) \in \mathbb{R}^{n \times m}$

We then compute the summaries of the context CQ in consideration of each word of the question as $C^Q = DA^Q \in \mathbb{R}^{\ell \times n}$.

We also compute the summaries of the question in consideration of each word of the context as $C^{D_1} = QA^D \in \mathbb{R}^{\ell \times m}$. Additionally, we compute the summaries of the previous attention contexts in consideration of each word of the context as $C^{D_2} = C^Q A^D \in \mathbb{R}^{\ell \times m}$. We then concatenate C^{D1} with C^{D2} along the row axis to form the coattention context $C^D \in \mathbb{R}^{2\ell \times m}$.

Finally, a bidirectional LSTM is used to integrate the temporal information with the coattention context as $u_t = Bi-LSTM\left(u_{t-1}, u_{t+1}, \left[d_t; c_t^D\right]\right) \in \mathbb{R}^{\ell}$. The resulting coattention encoding matrix is defined as $U = \left[u_1, u_2, \cdots, u_m\right] \in \mathbb{R}^{\ell \times m}$, and serves as the foundation to select the best possible answer span.

2.3 Multilayer Feedforward Neural Network Decoder

A multilayer feedforward neural network is utilized as a decoder to identify the start and end indices of the answer span. Our multilayer decoder is illustrated in Figure 2. The decoder mechanism is different to that used in (Xiong et al., 2016). The reference DCN implementation uses a Highway Maxout Network (HMN) based iterative decoder to find the answer span (Srivastava, Greff, & Schmidhuber, 2015).

We initially explored a naïve decoder that performs a linear mapping of the coattention encoding matrix U to produce score vectors $U1_s$ and $U1_e$, which contains the intermediate scores of each context word as the start and end indices respectively. Both $U1_s$ and $U1_e$ have the general linear mapping form of $U_1 = UW_0 + b_0$ where $U_1 \in \mathbb{R}^{m \times 1}$ $W_0 \in \mathbb{R}^{\ell \times 1}$ $b_0 \in \mathbb{R}^{m \times 1}$, but with an independent set of parameters $W0_s$, $b0_s$, $W0_e$, and $b0_e$. The score vectors $U1_s$ and $U1_e$ are converted into the corresponding probability vectors P_{start} = *softmax*(exp_mask($U1_s$, context_mask)) and P_{end} = *softmax*(exp_mask($U1_e$, context_mask)). The exponential masking operation, exp_mask, and the context mask vector, context_mask, are used to account for the padded words. They are described in further detail in section 2.4.

The naïve decoder produces intermediate scores for the start and end index scores from the coattention encoding matrix U in which each context word is considered in isolation. There is no contribution from the encoding of other words in the context towards the score of a given word. This motivated us to implement a more complex decoder that accounts for the interactions between the individual context word encodings.

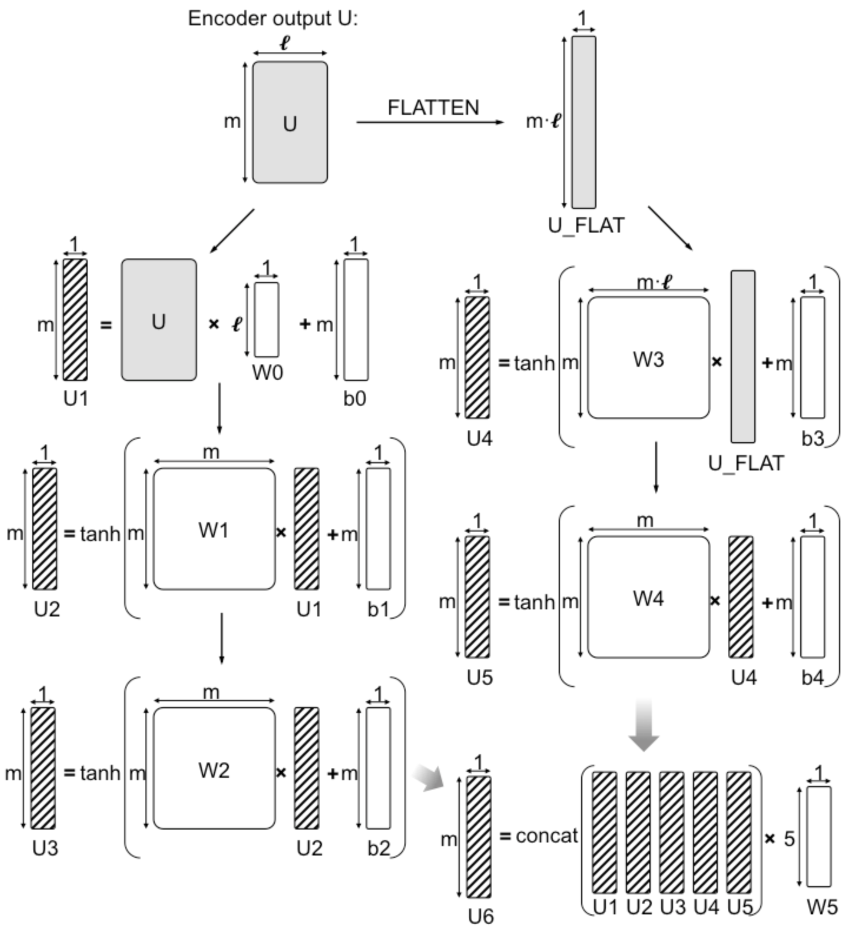

Figure 2: Multilayer feedforward neural network decoder. One model is used to calculate vector $U6_s$, which contains the intermediate scores of each context word as the start index. Another model is used to calculate vector $U6_e$, which contains the intermediate scores of each context word as the end index. The probabilities for the start and end indices are then obtained from these scores. Each model uses an independent set of parameters W0, b0, W1, b1, W2, b2, W3, b3, W4, b4, W5.

We implemented two fully connected layers with tanh nonlinearity to capture the interactions across word encodings for both the start and end indices. We picked two layers because (Xiong et al., 2016) reported decent results with a two layer MLP decoder. The general forms of these connected layers are:

$U_2 = \tanh\left(W_1 U_1 + b_1\right)$ where $U_2 \in \mathbb{R}^{m\times 1}$ $W_1 \in \mathbb{R}^{m\times m}$ $b_1 \in \mathbb{R}^{m\times 1}$

$U_3 = \tanh\left(W_2 U_2 + b_2\right)$ where $U_3 \in \mathbb{R}^{m\times 1}$ $W_2 \in \mathbb{R}^{m\times m}$ $b_2 \in \mathbb{R}^{m\times 1}$

Additionally, since information is condensed when U is transformed to U1, we also define a fully connected layer with tanh nonlinearity operating on the flattened version of U, U_{flat} that has access to the full context encoding. The general form of this connected layer is $U_4 = \tanh\left(W_3 U_{flat} + b_3\right)$ where $U_4 \in \mathbb{R}^{m\times 1}$ $W_3 \in \mathbb{R}^{m\times m\ell}$ $U_{flat} \in \mathbb{R}^{m\ell\times 1}$ $b_3 \in \mathbb{R}^{m\times 1}$. We also implement an additional layer with the general form of $U_5 = \tanh\left(W_4 U_4 + b_4\right)$ where $U_5 \in \mathbb{R}^{m\times 1}$ $W_4 \in \mathbb{R}^{m\times 1}$ $b_4 \in \mathbb{R}^{m\times 1}$.

Finally, we concatenate the vectors U1, U2, U3, U4, U5 and then apply a learnable weight vector W5 that automatically sets the relative importance of each vector in obtaining the final score vector $U6_s$ and $U6_e$ for either start and end indices respectively. $U6_s$ and $U6_e$ have the general form of $U_6 = \tanh\left[U_1, U_2, U_3, U_4, U_5 \right] W_5$ where $U_6 \in \mathbb{R}^{m\times 1}$ $W_5 \in \mathbb{R}^{5\times 1}$. Analogous to the naïve decoder, the score vectors $U6_s$ and $U6_e$ are converted into the corresponding probability vectors P_{start} = *softmax*(exp_mask($U6_s$, context_mask)) and P_{end} = *softmax*(exp_mask($U6_e$, context_mask)).

2.4 Loss Function

A fixed question length and fixed context length is enforced, so any question and context shorter than their respective fixed lengths are padded. As a result, the loss must be masked for answer words that have start and end indices that fall in the padded region. We employ the exponential mask function that adds a large negative number to scores that correspond to padded words:

exp_mask(scores, context_mask) = scores + (context_mask - 1)($10e^{-32}$), where context_mask entries are 0 for padded words and 1 for context words.

The resulting loss function that is minimized is:

CE(softmax(exp_mask(start_score)), ground_truth_start_index) + CE(softmax(exp_mask(end_score)), ground_truth_end_index)

Additionally, L2 regularization for the decoder weights is implemented.

2.4 Answer Span Prediction

We explored two approaches to predict answer spans based on the P_{start} and P_{end} vectors. The independent prediction approach predicts the start and end indices of the answer span independently using:

start_index = *argmax*(P_{start}) and end_index = *argmax*(P_{end})

The joint prediction approach predicts the start and end indices of the answer span to be the pair of indices that has the largest sum of the start and end probabilities among all the legal start and end indices pairs, where end_index >= start_index:

3 Related Work

Since the publication of the SQuAD dataset, there has been significant progress in applying neural network based models to the QA task. In particular, neural network based models have been shown to be particularly suited to the relatively complicated answers in the SQuAD dataset, which can be long phrases and often include non-entities.

(Wang & Jiang, 2016) proposed an end-to-end neural network model which consists of a Match-LSTM encoder (Wang & Jiang, 2015), and a pointer network decoder (Vinyals, Fortunato, & Jaitly, 2015). (Yu et al., 2016) proposed a dynamic chunk reader, which is a neural network based model that extracts a set of variable length answer candidates from the context and ranks them to answer the question. (Lu, Yang, Batra, & Parikh, 2016) proposed a hierarchical coattention model for visual question answering where the coattention mechanism simultaneously encodes a conditional representation of the image given a question as well as a conditional representation of the question given the image. (Xiong et al., 2016) proposed a dynamic coattention model (DCN) which consists of a coattentive encoder and a novel dynamic decoder that iteratively updates the start and end indices of the answer span.

Our model is heavily inspired by the DCN model, however we use a novel multilayer feed forward neural network decoder that calculates the probabilities of all the possible answer start and end index pairs in the context in a single pass, and picks the highest probability index pair as the final answer span.

4 Experiments

4.1 Implementation

Our model is trained and evaluated on the SQuAD dataset. The corpus is preprocessed using the Stanford CoreNLP tokenizer (Manning et al., 2014). We experimented with both fixed CommonCrawl.840B.300d pretrained word vectors and GLoVE.6B.100d pretrained word vectors (Pennington, Socher, & Manning, 2015)

We enforce a fixed question length of 22 words, and fixed context length of 300 words. Any question and context longer than their respective fixed lengths are trimmed and those shorter are padded up their respective max lengths. Overall this resulted in 98.9% questions and 98.35% contexts remaining in the training set.

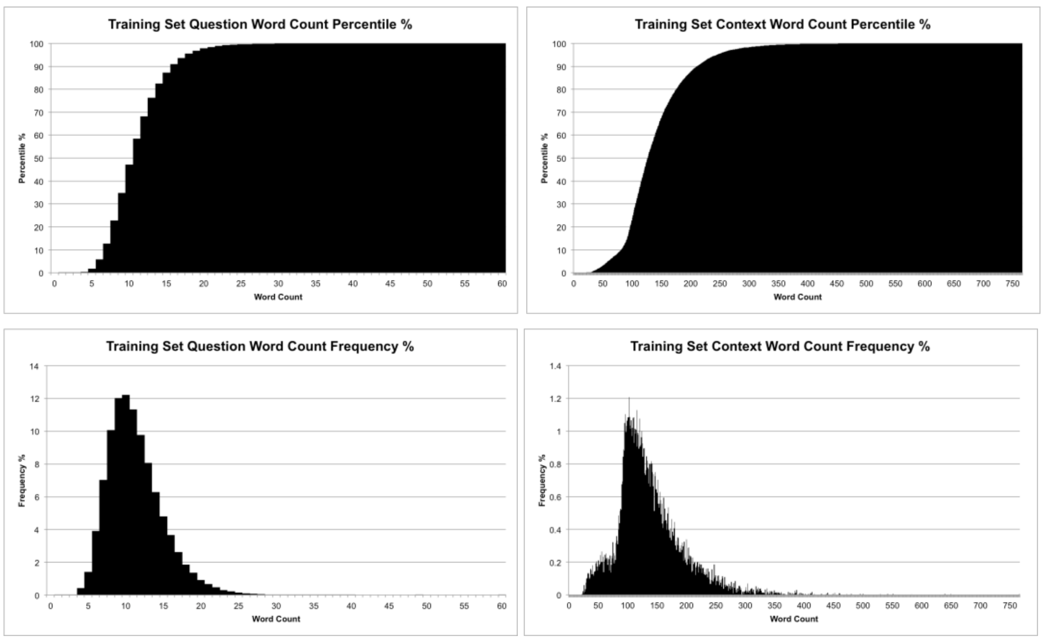

Figure 3: Training dataset statistics

Table 1: Hyperparameters used in our model

Hyperparameters	Value
Learning rate	0.002 → 0.0008 with exponential decay
Gradient clipping	5
Dropout (P_{keep})	0.85
L2 regularization	0.01
Batch size	32
Hidden state size	140
Fixed question size	22
Fixed context size	300

All models were implemented and trained with Tensorflow v0.12 (Abadi et al., 2015).

4.2 Results

We utilize the same metrics that were introduced in the original SQuAD publication: Exact Match (EM) and F1 score. The Exact Match score measures the percentage of predictions that match one of the ground truth answers exactly. The F1 score measures the average overlap between the prediction and ground truth answer. We consider the prediction and ground truth as a bag of words to compute their F1 score. Since a context question pair can have multiple ground truth answers, we take the maximum value of the EM and F1 across all the ground truth answers for a given question. We then compute the average over all the context question pairs to obtain the overall Exact Match and F1 scores.

The performance of our model on the SQuAD test dataset compared with the current top 4 submitted single models on the SQuAD leaderboard, and also the Dynamic Coattention Networks model, is shown in Table 1. Our single model results in a 52.8% Exact Match and 64.5% F1 on the test set.

Table 2: Performance comparison of our model with the current SQuAD single model leaderboard. * indicates unpublished model

Rank	**Model**	**EM**	**F1**
4	r-net*	72.4	80.8
6	Ruminate Reader*	70.6	79.5
7	ReasoNet*	70.6	79.4
7	Document Reader*	70.7	79.4
13	Dynamic Coattention Networks (Xiong et al., 2016)	66.2	75.9
	Our Model	**52.8**	**64.5**

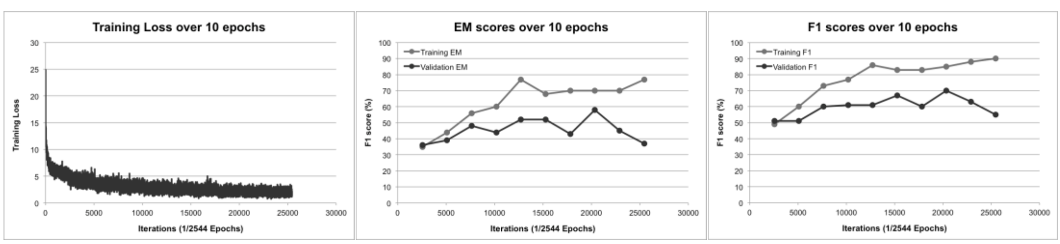

Figure 4: Training statistics. Our model starts to overfit after epoch 8

We found that the following factors significantly improved the performance of the model. The performance scores are the maximum F1 scores over 10 epochs on our validation dataset.

- No masking versus exponential masking for padded words masking (**F1 6% to 10%**).
- GloVE.6B.100d pretrained word vectors versus CommonCrawl.840B.300d pretrained word vectors (**F1 10% to 28%**).
- Answer span selection based on independently choosing the maximum probability start and end indices versus choosing the joint sum probability of the start and end indices **(F1 28% to 35%**)
- Naïve decoder versus multilayer decoder on multiple representations of the encoder outputs (**F1 35% to 67%**).

4.2 Error analysis

Apart from obviously incorrect answers that do not answer the question, there are other types of errors that are not completely wrong. The SquAD dataset ground truth answers were obtained from crowd sourced human annotations. As a result, it is almost certain that some ground truth answers are suboptimal, and other equivalent or better answers are possible for some questions. Thus, there could be multiple possible answers that, although do not exactly match the ground truth answer, is for all intents and purposes correct in answering the question.

- One type of frequently encountered errors is when the predicted answer span is narrower than the ground truth. In most cases, these predictions are functionally equivalent to the ground truth answers.

  Pred: " Nike advertisement"
  Truth: " a Nike advertisement"

- A related type of error is when the predicted answer span is wider than the ground truth. Sometimes the prediction is not specific enough, while othertimes the prediction is functionally equivalent to the ground truth answers

  Pred: "The minority leader , in consultation with other party colleagues , has a range of strategic options that he or she can employ to advance minority party objectives"
  Truth: " in consultation with other party colleagues"

- Another type of error is when the predicted answer span does not overlap with the ground truth answer span, yet the predictions are are functionally equilvalent.

  Pred: "OECD"
  Truth: "Organisation for Economic Co-operation and Development"

- Some errors should not be considered errors. Instead the predicted answers are better and clearer than the ground truth answers.

  Question: To what gods did Valerian tell the Christians to sacrifice ?
  Pred: "Rome 's traditional gods"
  Truth: "Rome 's traditional"

5 Conclusion

Overall, we propose a model that consists of a coattention encoder which learns codependent representations of the question and the context, and a novel multilayer feed forward neural network decoder that estimates the answer span in a single pass. On the SQuAD test dataset, our model achieves a single model performance of 52.8% EM and 64.5% F1. In the future, we will analyze the effects of ensembling on the model performance. We will also explore adding an LSTM to the decoder in order to select the start and end indices from the final probability vectors. Additionally, we will perform additional hyperparameter searching, such as modifying the fixed length question and context cutoffs.

Acknowledgments

We thank the CS224N teaching staff for their helpful comments and insights.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Research, G. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved from http://download.tensorflow.org/paper/whitepaper2015.pdf

Lu, J., Yang, J., Batra, D., & Parikh, D. (2016). Hierarchical Question-Image Co-Attention for Visual Question Answering. Retrieved from http://arxiv.org/abs/1606.00061

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & Mcclosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. Retrieved from https://nlp.stanford.edu/pubs/StanfordCoreNlp2014.pdf

Pennington, J., Socher, R., & Manning, C. D. (2015). GloVe: Global Vectors for Word Representation, 1532–1543. Retrieved from http://www.aclweb.org/anthology/D14-1162

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. Retrieved from http://arxiv.org/abs/1606.05250

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training Very Deep Networks. Retrieved from http://arxiv.org/abs/1507.06228

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer Networks. Retrieved from http://arxiv.org/abs/1506.03134

Wang, S., & Jiang, J. (2015). Learning Natural Language Inference with LSTM. Retrieved from http://arxiv.org/abs/1512.08849

Wang, S., & Jiang, J. (2016). Machine Comprehension Using Match-LSTM and Answer Pointer. Retrieved from http://arxiv.org/abs/1608.07905

Xiong, C., Zhong, V., & Socher, R. (2016). Dynamic Coattention Networks For Question Answering. Retrieved from http://arxiv.org/abs/1611.01604

Yu, Y., Zhang, W., Hasan, K., Yu, M., Xiang, B., & Zhou, B. (2016). End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension. Retrieved from http://arxiv.org/abs/1610.09996