Neural Network-based Question Answering System

Kushaagra Goyal
Department of Electrical Engineering
Stanford University
goyalk@stanford.edu

Sriraman Madhavan
Department of Statistics
Stanford University
sriraman@stanford.edu

Sanyam Mehra
Department of Electrical Engineering
Stanford University
sanyam@stanford.edu

Abstract

The idea of a Question Answering (QA) system is to extract information (sometimes passages, or spans of words) directly from documents, conversations, online searches, etc., that will meet the user's information needs. In this work, we focus on the Stanford Question Answering Dataset (SQuAD) and propose an end-to-end deep neural network model for machine comprehension, while achieving an F1 score of 61.13% and an Exact Match (EM) score of 46.92% on the test dataset.

Codalab submission username: *goyalk*
F1 score on test dataset: *61.13%*
EM score on test dataset: *46.92%*

1 Introduction

Building accurate Question-Answering systems is a compelling yet challenging task in natural language processing research. A QA system can combine easily with other NLP systems, like chatbots. Some QA systems even go beyond the search of text documents and can extract information from a collection of pictures. However, the limited size of previously available datasets prevented researchers from building end-to-end deep neural network models. To address this weakness, Rajpurkar et al. (2016) [1] developed the Stanford Question Answering dataset (SQuAD). SQuAD comprises around 100K question-answer pairs, along with a context paragraph. The context paragraphs were extracted from a set of articles from Wikipedia. Humans generated questions using that paragraph as a context, and selected a span from the same paragraph as the target answer. In this task, answering a question is defined as predicting an answer span within a given context paragraph.

2 Task Definition

The task can be represented as estimating the conditional probability Pr(A/Q,P) from the training set and predicting answers for testing instances by:

$$A^* = argmax_{A \in A(P)}(Pr(A/Q,P)) \tag{1}$$

A(P) contains all possible substrings of P. We make an independent assumption where we try to predict the starting and ending indices of the answer span instead. [2]

$$A^* = argmax_{0<=a_s<=a_e<P} Pr(a_e/Q,P)Pr(a_s/Q,P) \tag{2}$$

3 Related Work

Wang et al. [2] propose a Multi-Perspective Context Matching (MPCM) model, which is an end-to-end system that directly predicts the answer beginning and ending points in a passage, by forming a matching vector from multiple perspectives. Xiong et al. [3] introduce the Dynamic Coattention Network (DCN) for question answering, which fuses co-dependent representations of the question and the document in order to focus on relevant parts of both. Seo et al. [4] introduce the Bi-Directional Attention Flow (BIDAF) network, a multi-stage hierarchical process that represents the context at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation without early summarization. In this project, we try to build a hybrid model that combines certain functionalities from the above state-of-the-art systems.

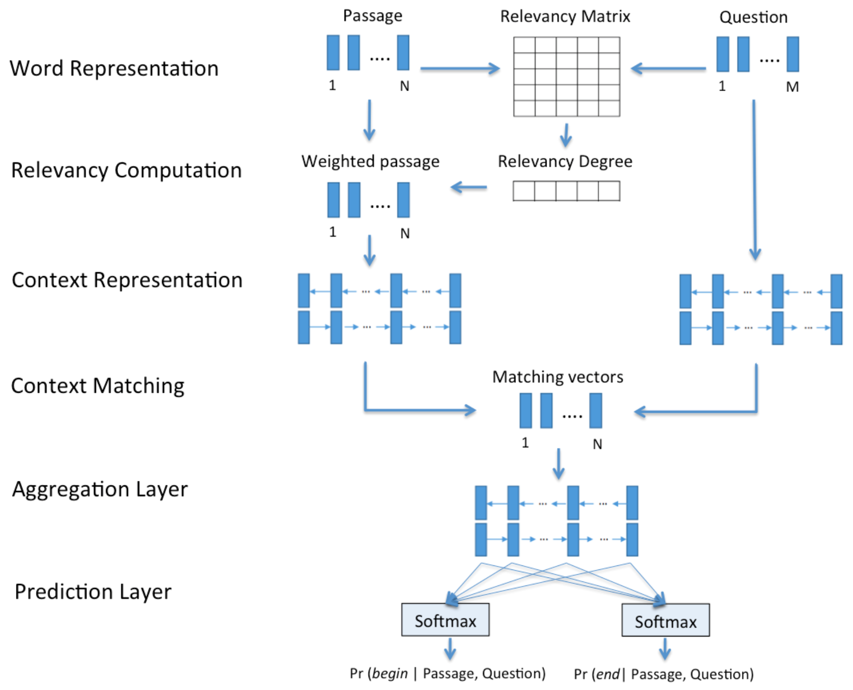

Figure 1: Multi-Perspective Context Matching Model Architecture

4 Approach

4.1 Multi-perspective Context Matching models

Our initial model encodes the question and passage by using bi-directional LSTMs, while incorporating the relevancy of each word in the passage to the question, in the encoding process. For each point in the passage, the model then matches the context of this point against the encoded question from multiple perspectives and produces a matching vector. Given those matched vectors, the model employs another bi-directional LSTM to aggregate all the information and predict the beginning and ending points. We discuss each step in detail, in the following subsections.

4.1.1 Word representation

We represent each word in the question and passage with a 100-dimensional vector. The embeddings are pre-trained with GloVe (Pennington et al., 2014). The output of this layer is word vector sequences for question Q : $[q_1, ..., q_M]$, and passage P : $[p_1, ..., p_N]$.

4.1.2 Relevancy Computation

We compute a relevancy matrix which quantifies the relevancy between every word in the passage and every word in the question. The relevancy is computed by simply taking the cosine similarity of the embeddings of the words involved: $r_{i,j} = \frac{{q_i}^T p_j}{\|q_i\| \|p_j\|}$. Then, the *relevancy degree* of each passage word is the maximum relevancy among all computed quantities involving the word: $r_j = max\{r_{i,j}\}$. The relevancy degrees are then multiplied with the word embeddings before moving on to the next layer $\mathbf{p}'_j = r_j \cdot \mathbf{p}_j$ to ensure that relevant words are given a higher weight in the subsequent layers, than irrelevant and common words like stop words, etc. The output of this layer is word vector sequences for question Q : $[q_1, ..., q_M]$, and weighted passage P' : $[{p_1}', ..., {p_N}']$.

4.1.3 Context Representation

In this layer, we incorporate contextual information into the way the question and passage are represented. We use a bidirectional LSTM to encode contextual embeddings for each question word and passage word.

$$\begin{aligned}
\vec{h_i}^q &= L\vec{ST}M(\vec{h_{i-1}}^q, q_i) \qquad i = 1, ..., M \\
\overleftarrow{h_i}^q &= L\overleftarrow{ST}M(\overleftarrow{h_{i-1}}^q, q_i) \qquad i = 1, ..., M \\
\vec{h_j}^p &= L\vec{ST}M(\vec{h_{j-1}}^p, p_j) \qquad j = 1, ..., N \\
\overleftarrow{h_j}^p &= L\overleftarrow{ST}M(\overleftarrow{h_{j-1}}^p, p_j) \qquad j = 1, ..., N
\end{aligned}$$

4.1.4 Context Matching

In this layer, we compare the constructed contextual embeddings of the passage to the query from multiple perspectives. We implemented three matching strategies to compare each contextual embedding of the passage to the question, namely *Full matching, Max-pooling matching,* and *Mean-pooling matching*, as described in Wang et al. [2].

4.1.5 Aggregation Layer

This layer involves another bidirectional LSTM on the matching vectors that were generated in the previous step. This BiLSTM facilitates interactions at each time step of the passage. The aggregation vectors that are generated, are used to predict the answer span, in the next layer.

4.1.6 Prediction Layer

We have implemented a feedforward neural network to predict the probability distributions for the span-start and span-end positions. We feed the aggregation vector of each time step into the fully connected layer individually, calculate a value for each time step, then normalize the values across the entire passage with softmax operation.

4.1.7 Results

The model's performance was tested on the validation set, and it did not perform as expected, and had a very low performance. The results obtained from this model are shown in table 1.

Table 1: Model Performance on val set

Model	**F1 (%)**	**EM(%)**
Full Matching only	28.2	20.4
All Matchings	32.1	23.3

We concluded that the joint representation of the passage from this model was not being learned effectively, which is why we decided that implementing a co-attention mechanism will probably improve the model performance.

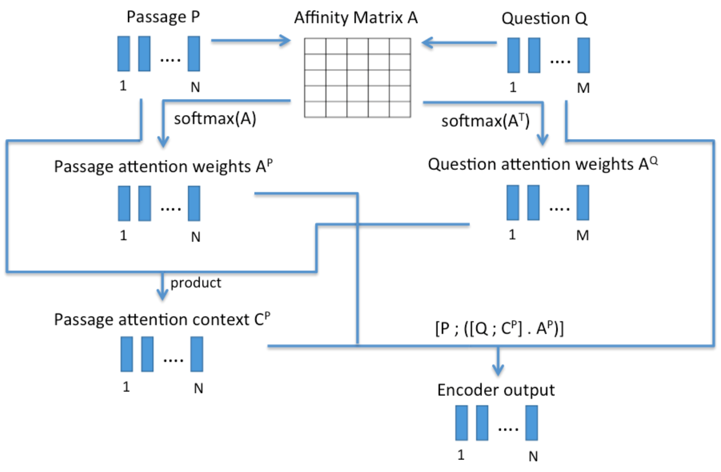

Figure 2: Co-attention Encoder Architecture

4.2 Co-attention Models

4.2.1 Naive Model

We implemented a simple co-attention mechanism (similar to relevancy computation) after the Context Representation Layer, that attends to the question and document simultaneously and finally fuses both attention contexts. The resulting representation is fed to a simple Bi-LSTM and softmax layer to compute the probability distributions of span start and span end.

4.2.2 Co-Attention Mechanism

We then proceeded to improve the encoding mechanism in the naive model. Figure 2 describes the steps involved in incorporating the co-attention mechanism into the passage representation. We first compute the affinity matrix which contains the affinity scores of all pairs of passage and question words: $A = P^T Q$. The affinity matrix is then normalized row-wise to produce the attention weights across the document for each word in the question, and column-wise to produce the attention weights across the question for each word in the document.

$$A^P = softmax(A); \quad A^Q = softmax(A^T) \tag{3}$$

The resulting weights are then combined with the initial representations to give a final encoder output with co-attention, as described in Xiong et al. [3].

$$(Passage\,Attention\,Context)\; C^P = P \cdot A^Q \tag{4}$$
$$Final\,Encoder\,Output = [P; ([Q; C^P] \cdot A^P)] \tag{5}$$

4.2.3 Context Modeling

We then proceeded to improve the decoder of the intermediate model by adding a layer that employs a Recurrent Neural Network to scan the context. The co-attention encoder output is fed into a double-layered Bi-LSTM, as described in Seo et al. [4]. We observe that this allows to decode the joint representation of the question and the context, subsequently to compute the probability distributions of *span start* and *span end*. The steps in the decoder architecture are shown in Figure 3.

4.3 Experiment Details

We use 100 dimensional GloVe vectors and limit the vocabulary to the words in the training set. We set the maximum passage length to 300, this helps in speeding up the training. Out of 87k examples,

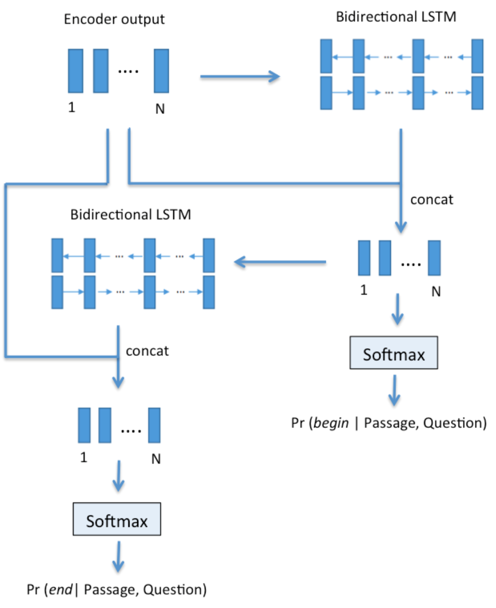

Figure 3: Decoder architecture

Table 2: Coattention Models' Performance on Dev set

Model	**F1 (%)**	**EM (%)**
Naive Model	33.6	24.2
+ Co-Attention Mechanism	45.2	34.5
+ Context Modeling Layer	60.4	45.5

$\sim$ 1500 samples had a passage length of greater than 300. These small fraction of samples were removed from training. We limited the question length to 25, again so that the small fraction of such questions are automatically truncated. We trained our model with various dropout values ranging from 0.05 to 0.5 and also with different hidden state sizes 50, 100, 150. We observed the best performance with the state size of 150 and a dropout of 0.1. We ran our model for 4 epochs and each epoch takes $\sim$ 5 hours of training time on the GPU. The model was optimised using the ADAM optimiser with a learning rate of 0.001.

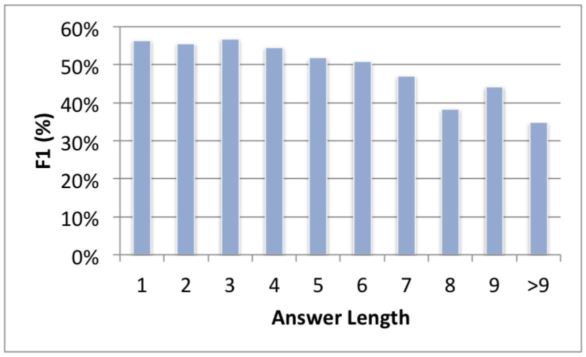

Figure 4: Performance Variation with Answer Length

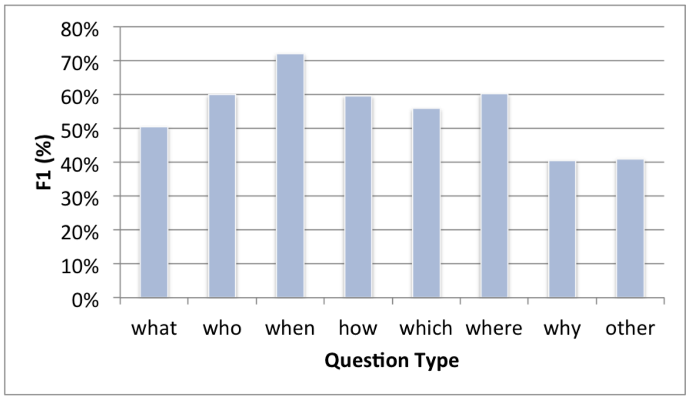

Figure 5: Performance Variation with Question Type

5 Results

Evaluation of the model is done with two metrics, namely F1 and EM score. The EM (Exact Match) calculates the exact string match between the predicted and the ground truth answer whereas F1 calculates the overlap between the words in predicted and the actual answer. The results of our variations of the co-attention models are shown in Table 2, and the performance of the final model is shown in Table 3.

Table 3: Final Model Performance on SQuAD

Data set	**F1 (%)**	**EM (%)**
Training set	74.5	59.2
Dev data set	60.4	45.5
Test data set	61.1	46.9

We analysed the variation in performance of our model with the answer length. 5 We observed that with increasing question length, the performance decreases. But this is very intuitive, since longer answer spans will be much more challenging to compute.

We analysed the variation of the performance with the different question types in 4. We observe that 'when' type questions have the maximum performance whereas, the 'why' type questions have a lower performance. Our models struggles with the 'why' questions, because these are complex and require a deeper level of reasoning to answer. Also, the answer to 'why' questions typically tend to be longer, and as seen in Figure 4, the performance of the model is lower.

6 Conclusion

We implemented an end-to-end-neural network architecture for question answering.

Our model identifies the answer span by matching each time-step of the passage with the question using a co-attention encoder which learns the co-dependent representations of the question and the document. The bi-directional attention mechanism feeds query-aware context representation to predict the beginning and ending points based on globally normalizing probability distributions. Future work involves exploring linear, bilinear and linear with perceptron attention. Specific model enhancements to improve performance on 'why', 'what', 'which' type of questions can also lead to a better QA system.

References

[1] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

[2] Wang, Zhiguo, et al. ”Multi-Perspective Context Matching for Machine Comprehension.” arXiv preprint arXiv:1612.04211 (2016).

[3] Xiong, Caiming, Victor Zhong, and Richard Socher. ”Dynamic Coattention Networks For Question Answering.” arXiv preprint arXiv:1611.01604 (2016).

[4] Seo, Minjoon, et al. ”Bidirectional Attention Flow for Machine Comprehension.” arXiv preprint arXiv:1611.01603 (2016).

[5] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. ”Glove: Global Vectors for Word Representation.” EMNLP. Vol. 14. 2014.

[6] Wang, Shuohang, and Jing Jiang. ”Machine comprehension using match-lstm and answer pointer.” arXiv preprint arXiv:1608.07905 (2016).

[7] Ittycheriah, Abraham, et al. ”IBM’s Statistical Question Answering System.” TREC. 2000.

[8] Zhang, Junbei, et al. ”Exploring Question Understanding and Adaptation in Neural-Network-Based Question Answering.” arXiv preprint arXiv:1703.04617 (2017).