Machine Comprehension with Deep Learning on SQuAD dataset

Yianni D. Laloudakis
Department of Computer Science
Stanford University
jlalouda@stanford.edu
CodaLab: jlalouda

Neha Gupta
Department of Computer Science
Stanford University
nehgupta@stanford.edu
CodaLab: nehagupta

Yash H. Vyas
Department of Statistics
Stanford University
yashvyas@stanford.edu
CodaLab: yashvyas(submitter)

Abstract

In a world with too much information and not enough time to parse it, humans could greatly benefit from an automated question answering system. End-to-end models that answer questions based on a passage would have profound impact on people inquiring about medical, legal, or technical documents. We choose to address this problem using deep learning techniques, that have recently come onto the scene and vastly outperform traditional machine learning models in language modeling. This project aims to replicate two of the leading papers in the domain of automated question answering and report on the results ([6], [4]).

1 Introduction

Machine comprehension, specifically automated question answering, poses one of the greatest challenges to the AI and NLP communities. Traditional approaches to tackling this challenge involved human designed features. However, much progress has been made by transitioning from classification based on hand-crafted features to end-to-end deep learning models [1], [2]. Existing state-of-the-art methods are able to capture the contextual information using memory networks and attention mechanisms. The goal of this project is to implement two such methods that utilize pointer networks and bidirectional attention flow ([5] and [3]).

2 Dataset

This project uses the Stanford Question Answering Dataset (SQuAD) to train two end-to-end models for automated question answering. The train set consists of about 80,000 context, question, and answer triples that are used to train our model. The answer is a continuous sequence of words within the context paragraph, and the objective is to predict this answer.

2.1 Motivating Examples

This is a typical example of a context, question, answer triple in the dataset.

Ex1: Context:

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

The iPod has also been credited with accelerating shifts within the music industry . The iPod's popularization of **digital music storage** allows users to abandon listening to entire albums and instead be able to choose specific singles which hastened the end of the Album Era in popular music.

Question: What did the iPod promote that prompted a big change in the music industry?

Answer: digital music storage

Ex2: Context:

The prevalence of HIV-infection among the adult population is **1.8%**. Only 20% of infected pregnant women receive anti retroviral coverage to prevent transmission to newborns

Question: What percentage of the adult population has an HIV-infection?

Answer: 1.8%

Ex3: Context:

Since 1934 , the sale of Federal Duck Stamps has generated **$670 million** , and helped to purchase or lease 5,200,000 acres (8,100 sq mi ; 21,000 km2) of habitat . The stamps serve as a license to hunt migratory birds , an entrance pass for all National Wildlife Refuge areas , and are also considered collectors items often purchased for aesthetic reasons outside of the hunting and birding communities . Although non-hunters buy a significant number of Duck Stamps , eighty-seven percent of their sales are contributed by hunters , which is logical , as hunters are required to purchase them . Distribution of funds is managed by the Migratory Bird Conservation Commission (MBCC).

Answer: $670 million

How much money has been generated due to the Duck Stamp act ?

2.2 Data Structure

Our first step in processing the data was to plot histograms of the context, question, and answer lengths, which would affect our padding and truncation and our eventual train runtime. These graphs are shown below.

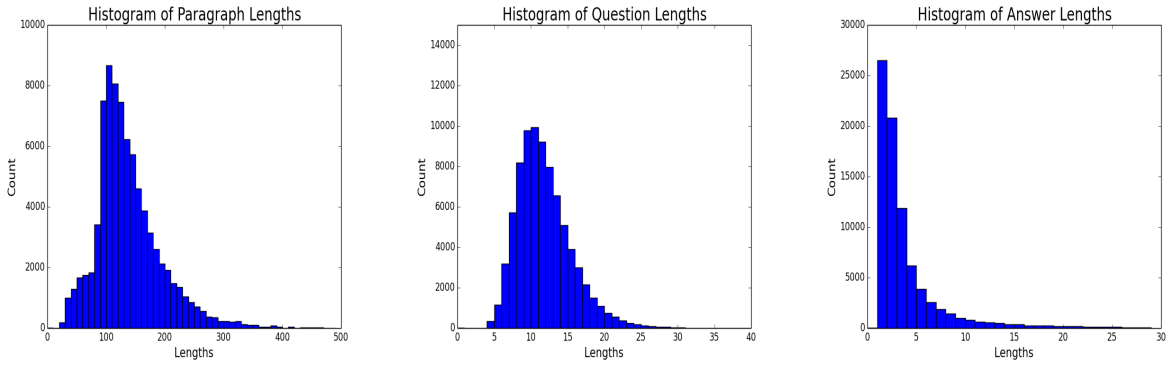

Figure 1: Left: Paragraph Lengths Middle: Question Lengths Right: Answer Lengths

Looking these histograms, we decided to truncate any paragraph with more than 300 words, any question with more than 30 words, and we only searched for answers of up to 20 words. These thresholds were selected so as to not affect more than 1% of the data while also prioritizing training speed.

2.3 Problem Statement

Since all the answers in the dataset are contiguous, we model the problem as predicting the start and the end index of the answer in the paragraph.

We minimize over the following cross-entropy loss function during training:

$$CE = -\frac{1}{N}\sum_{i}^{N} log(p^{s}_{ys_i}) + log(p^{e}_{ye_i})$$

where N is the total number of observation in train set, $p^{s}_{y_i}$ is the probability of the actual word ys_i being the start index and $p^{e}_{ye_i}$ is the probability of the actual word ye_i being the start index in the ith observation.

3 Model 1: Match-LSTM with Answer pointer

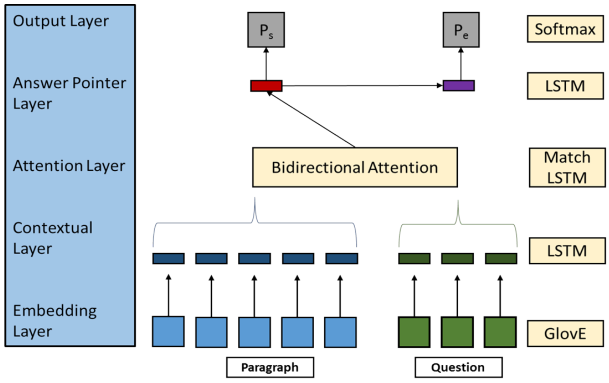

Figure 2: Schematic diagram of the Match-LSTM with Answer pointer (Boundary model)

The unique feature of Match-LSTM [6] architecture is that the model integrates calculations for the attention layer within an LSTM cell that traverses the context from both directions.

3.1 Embedding Layer

We used 100-dimensional GloVe vectors which were pretrained on Wikipedia text were used to represent the context and question. These vectors were not trained to reduce the number of parameters in the model and prevent overfitting to the train set.

3.2 Contextual Layer

A unidirectional LSTM is run over the context and question to produce $\mathbf{H} \in \mathbb{R}^{TxS}$ and $\mathbf{U} \in \mathbb{R}^{QxS}$ where T is the paragraph length, Q is the question length, and S is the state size, which we chose to be 200.

3.3 Attention Layer

Each of the T h_i's of the paragraph is compared to the Q u_i's of the question and attention result is then integrated into an LSTM computation. The equations are summarized below:

$$\overrightarrow{G_i} = \tanh(UW^q + (h^p_{i-1}W^p + \overrightarrow{h}^r_{i-1}W^r + b^p)) \tag{1}$$

$$\overrightarrow{a_i} = \text{softmax}(\overrightarrow{G_i}w + b) \tag{2}$$

Where $W^q, W^p, W^r \in \mathbb{R}^{SxS}$, $b^p \in \mathbb{R}^{\mathbb{S}}$ and b is a scalar.

Note that at then end of computation, $\overrightarrow{a_i}$ is derived from a softmax and has Q elements, each representing a normalized attention score for the context word over each word in the question. The input to the next time step in the Match-LSTM is formed in this way:

$$\overrightarrow{z_i} = \begin{bmatrix} h_i^p \\ \overrightarrow{a_i} U \end{bmatrix} \tag{3}$$

And finally,

$$\overrightarrow{h}^r = \overrightarrow{LSTM}(\overrightarrow{z_i}, \overrightarrow{h}^r_{i-1}) \tag{4}$$

The same structure is used to build a Match-LSTM with the context reversed and the final outputs, $\overrightarrow{\mathbf{H}^r}$ and $\overleftarrow{\mathbf{H}^r}$ are concatenated together to form $\mathbf{H}^r$.

3.4 Answer Pointer Layer

This layer is responsible for using the attention weights generated in the previous layer to create a probability distribution for the start and end word by pointing back into the context paragraph.

$$\mathbf{F}_k = \tanh(\mathbf{H}^r \mathbf{V} + (h_{k-1}^a \mathbf{W}^a + b^a)) \tag{5}$$

$$\beta_k = \text{softmax}(\mathbf{F}_k v + c) \tag{6}$$

$$h_k^a = \overrightarrow{LSTM}(\beta_k \mathbf{H}^r, h_{k-1}^a) \tag{7}$$

Where $\mathbf{V} \in \mathbb{R}^{Sx2S}$, $\mathbf{W}^a \in \mathbb{R}^{SxS}$, $b, v \in \mathbb{R}^S$ and c is a scalar.

We run the LSTM twice to produce two β's, one for the start and end index, where β_i^1 is the probability that word i in the context is the beginning of the answer and β_j^2 is the probability that word j in the context is the end of the answer.

3.5 Output

A search is done over all combinations of start and end words in a window size of 20 to find the combination with the highest joint probability.

4 Model 2: BiDAF

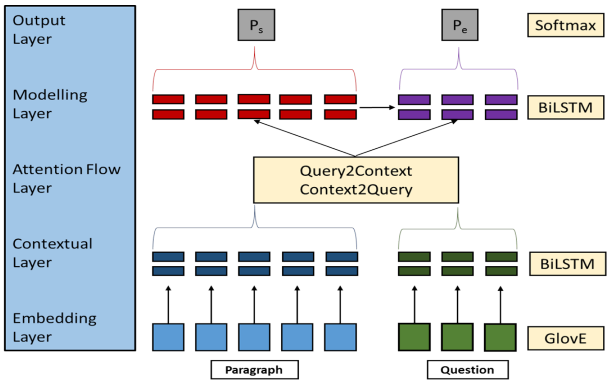

Figure 3: Schematic diagram of BiDAF model

4.1 Embedding Layer

100-dimensional GloVe vectors which were pretrained on Wikipedia text were used to represent the context and question. These vectors were not trained to reduce over-fitting to the train set.

We also wrote code for character embeddings but could not use it because it was not compatible with the version of TensorFlow on the GPU. The idea was as follows: We represented each character as a vector of some small dimension d and initialized it with random weights. If the word k has l character, then the character representation of k was given by the matrix,

$$C^k \in \mathbb{R}^{d \times l}$$

We then obtained a 100-dimensional embedding of each word by applying a convolution filter 100 times and then taking a non-linearity and then max over all the values of the filter. To be specific, each dimension of the word embedding was calculated as follows:

$$C'^k[h] = \max_i\{\tanh(\text{conv}(C, H_h) + b)\} \tag{8}$$

where H_h was chosen to be a filter of dimension $d \times w$ and w was chosen to be 5 in our case.

We then used a highway network to combine the word and the character embeddings. The character embeddings were trainable.

4.2 Contextual Layer

In the contextual layer, we used 2 forward and backward LSTMs on both paragraph and question strings to get $H \in \mathbb{R}^{2d \times T}$ from the paragraph vectors X and $U \in \mathbb{R}^{2d \times J}$ from the paragraph vectors Q.

4.3 Attention Flow

The Attention flow layer is used to represent the combined information from both the paragraph vectors and the question vectors. The inputs to the layer are the individual representations of the paragraph and the question vectors and this layer outputs the representation of the paragraph vectors in context of the question vector.

Here, S represents the similarity matrix between paragraph and question vectors. S_{tj}, indicates the similarity between the t'th paragraph word and the j'th question word.

$$S_{tj} = \alpha(H_{:t}, U_{:j}) \in \mathbb{R} \tag{9}$$

and α which encodes the similarity function is chosen in the following way:

$$\alpha(h, u) = w_s^T[h; u; h \circ u] \tag{10}$$

Next, we first compute which question words are most important for the paragraph words. Here, a_t represents the attention weights on the question words for the t'th paragraph word.

Thus, we know this:

$$\sum a_{tj} = 1 \tag{11}$$

The attention weight vector a_t is computed in the following manner:

$$a_t = \text{softmax}(S_{t:}) \tag{12}$$

Thus, the question vectors after applying the attention weights become:

$$\widetilde{U} = \sum_j a_{tj} U_{:j} \tag{13}$$

Now, we calculate the most important paragraph words in the context of the query words, which are the words most likely to appear in an answer. Similarly, we get the following equations where $\widetilde{H}$ represents the paragraph vectors weighted by their attention score:

$$b = softmax(max_{col}(S)) \tag{14}$$

$$\widetilde{h} = \sum_t b_t H_{:t} \tag{15}$$

We then concatenate the weighted vectors for paragraph and question along with the contextual representations to get G:

$$G_{:t} = \beta(H_{:t}, \widetilde{U_{:t}}, \widetilde{H_{:t}}) \tag{16}$$

where β is defined as follows:

$$\beta(h, \widetilde{u}, \widetilde{h}) = [h; \widetilde{u}; h \circ \widetilde{u}, h \circ \widetilde{h}] \tag{17}$$

4.4 Modeling

Then, the modeling layer is used to capture the temporal interactions between different paragraph words conditioned on the questions by using G as input. Here, we use two layers of bidirectional LSTM to get the output M.

4.5 Output

In the output layer, we predict the word of being a start and end word by multiplying the input by a weight vector w and computing the softmax.

$$p^1 = \text{softmax}(w_{p^1}^T [G; M]) \tag{18}$$

$$p^2 = \text{softmax}(w_{p^2}^T [G; M^2]) \tag{19}$$

Training loss is computed as the standard cross entropy loss over the prediction of start word index and end word index.

$$L(\theta) = -\frac{1}{N} \sum_i^N log(p_{y_i^1}^1) + log(p_{y_i^2}^2) \tag{20}$$

5 Results and Error Analysis

The data is split into train (95%) and validation (5%) sets from the total of 80,000 question, context, and answer triplets. We track our F1 and exact match (EM) scores to decide on the number of epochs and stop training when the scores on the validation set drops for three consecutive epochs. In the interest of time, we run a maximum of 12 epochs per run for each of the experiments performed.

5.1 Match LSTM Model

The best performance using the Match-LSTM model without using any dropout was with F1 score of 0.18 and EM as 0.098 (9.8%) on the dev set of the leader-board. The Match-LSTM paper [6] does not specify using dropout. However, taking cue from [4] and class assignments, we applied dropout of 0.15 for the linear transformations before the attention softmax layer. We only saw a marginal improvement in the F1 and EM scores. In all the experiments, we used the Adam optimizer with a learning rate of e-4.

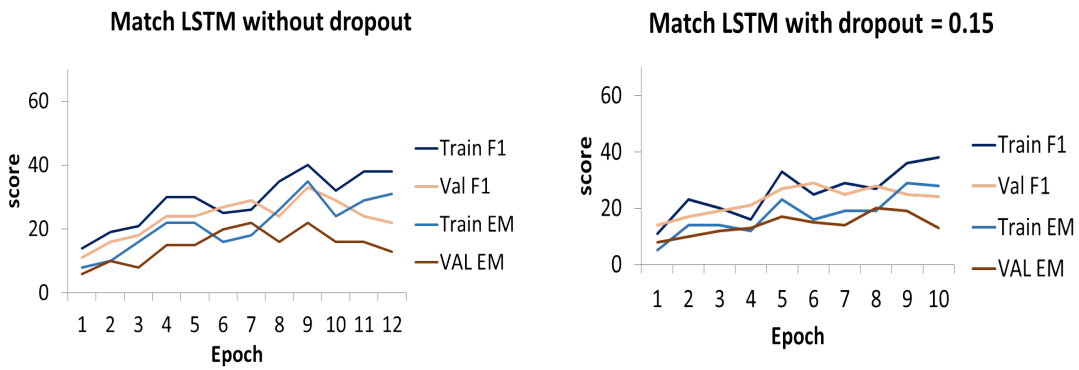

Figure 4: Results for LSTM (Boundary model)

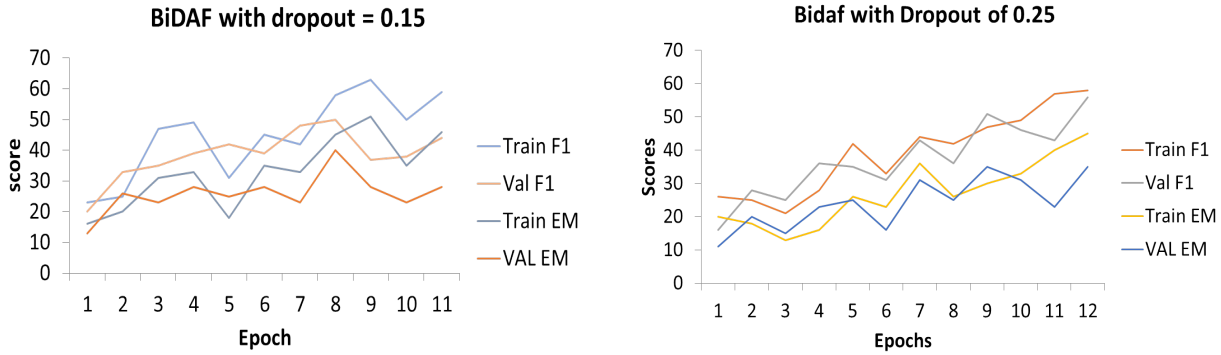

Figure 5: Results for BiDAF with dropout

From our observations on the validation set, we notice that the Match-LSTM model is able to predict the answer accurately when the answer span is less than or equal to 5. Compared to our overall best validation F1 score of 0.32 and EM of 0.22, we get an F1 score of 0.44 and EM of 0.34 on answers less than or equal to 5. Thus, we notice that longer answers are harder to predict.

The LSTM-Match also had low scores on "what", "why"and "which" type of questions. The reason for low scores on "why" and "how" is due to long answers associated with the question.

Besides, the model also fails on some of the single word answer.

Context:

Hunting big game typically requires a " tag " for each animal harvested . Tags must be purchased in addition to the hunting license , and the number of tags issued to an individual is typically limited . In cases where there are more prospective hunters than the quota for that species , tags are usually assigned by lottery . Tags may be further restricted to a specific area , or wildlife management unit . Hunting migratory waterfowl requires a duck stamp from the Fish and Wildlife Service in addition to the appropriate state hunting license.

Query: A wildlife management unit is a place where what may be restricted to ?

Answer: tags, Predicted:"specific area"

The model has not been able to distinguish between "where" and "what" in the question.

5.2 Bidaf Model

The BiDaf paper [4] uses a dropout of 0.2 across all CNN, LSTM and linear transformations. However, we experiment with dropout of 0.15 and 0.25 to see any variations. We increase our learning rate

from e-4 to e-3 and use Adam optimizer. We noticed that applying dropout of 0.15 gave better results on the dev set than with using dropout as 0.25. The results are shown in the figure.

The best performance using the Bidaf model using dropout of 0.15 was with F1 score of 0.49 and EM as 0.37 (37%) on the test set of the leader-board while dropout of 0.25 gave 0.45 and 0.34 as F1 and EM scores respectively.

The errors in Bidaf models were again higher for sentences with longer span. However, the model was able to single token answer with a very high level of accuracy. The accuracy on the "why" and "how" was again lower than other question types. For some examples, the paraphrase were not captured accurately.

Context:

The iPod has also been credited with accelerating shifts within the music industry . The iPod's popularization of digital music storage allows users to abandon listening to entire albums and instead be able to choose specific singles which hastened the end of the Album Era in popular music.

Question: What did the iPod promote that prompted a big change in the music industry?

Answer: digital music storage, Predicted: "specific singles"

6 Comparison of Models

We switched to the BiDAF model after poor results from match-LSTM. We can attribute BiDAF's success over Match-LSTM to three key causes: first, BiDAF computes attention in both directions, from context to question and question to context, while match-LSTM only computes attention in one direction. Second, BiDAF allows information from the contextual layer to flow through until the end of the model, while contextual representations are discarded by Match-LSTM as soon as they are done being used for computing attention. Third, BiDAF is overall a more complex model. Bi-directional LSTMS are used at all points while the match-LSTM authors report no increased benefit from using bidirectional LSTMs.

7 Leader board submission

The best performance on the test set used Bidaf model with dropout of 0.15 was with F1 score of 0.49 and EM as 0.37 (37%) on the test set of the leader-board. Results are submitted by yashvyas on behalf of the team.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. pages 3111–3119, 2013.

[3] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[5] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

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