Final default project: questions answering with deep learning

David Uvalle
SCPD Student
Stanford University
davidu@stanford.edu

Denis Ulanov
SCPD Student
Stanford University
dulanov@stanford.edu

Abstract

For this assignment we analysed properties of the SQuAD dataset, implemented several improvements over existing attention models, and finally we analysed the improvements' performance and errors as presented on this document. Our best model (ensemble) achieved an F1 score of 0.682 and EM of 0.563.

1 Introduction

Machine comprehension for Question Answering is an artificial intelligence task in which given a context and a question a program must produce an answer close to what a human would answer. The quality of the answers are evaluated by the F1 and Exact Match (EM) scores. Machine comprehension is still an open problem due to its complexity, for an instance, a model should be able to do coreference resolution and inference relatively well to compare to human performance. Stanford published a large questions and answers dataset (SQuAD) and a leaderboard where multiple research groups compete to create state of the art models with high F1/EM scores.

For this assignment we analyzed the data from SQuAD, implemented improvements to existing attention models, and analyzed their performance and errors. The rest of this document describes each of these activities in detail.

2 Problem definition

Given word embeddings of context K of length N, $K = \{K_1, K_2, \ldots, K_N\}$ and a question V of length M, $V = \{V_1, V_2, \ldots, V_M\}$ learn a function $f(K,V) = (A_S, A_E)$, where A_S and A_e represent the start and end positions of the answer of question V in the context K.

3 Model architecture

We started with the baseline model provided to us on the handoff and added several improvements that we describe in details in sections 5 and 6. The final model architecture is presented in Figure 1.

The input context words and question words come to an input layer where they get truncated to the max context/question length and get padded if they are shorter than the maximum. Embeddings layer converts the words to GLoVE word vectors. Feature extraction layer computes custom features such as POS tokens and exact/soft matches and concatenates them with word vectors. Encoder layer consists of a 1 layer bidirectional GRU unit. It's output is passed to a bi-directional attention layer, then to a ReLU unit. To get final predictions, we apply softmax to get the probabilistic distribution of start_pos and end_pos and then get the argmax over both distributions.

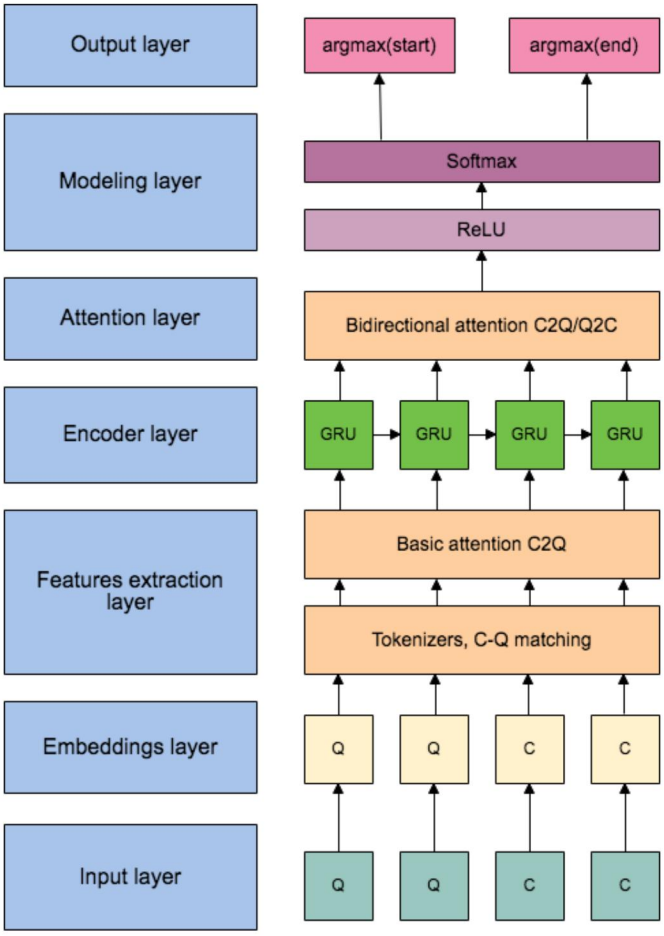

Figure 1. Model architecture

4 Input data analysis

Before starting the experiments, we looked at the aggregated statistics for the context/questions/answers triplets to get the intuition about the patterns that exist in input data and the most common scenarios that input data presents. We wanted to ensure that we aren't solving for use cases that don't exist in the training and dev datasets.

One of the metrics that we looked at is the length of context and questions, as well as the positions of the answer within the context. Figures 2, 3 and 4 show the distribution of context length, position of answer within the context and question length respectively.

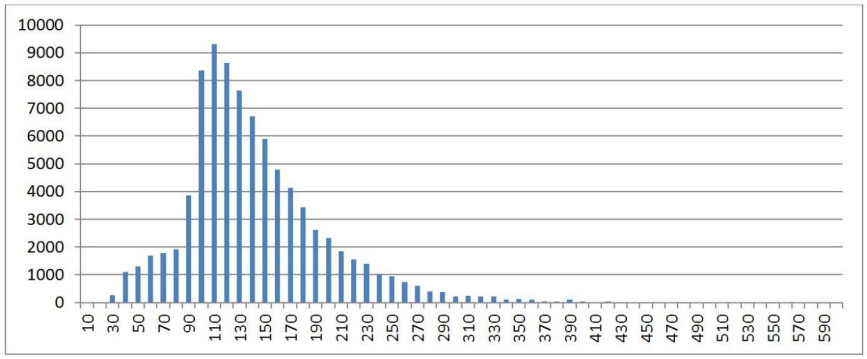

Figure 2. Context length distribution

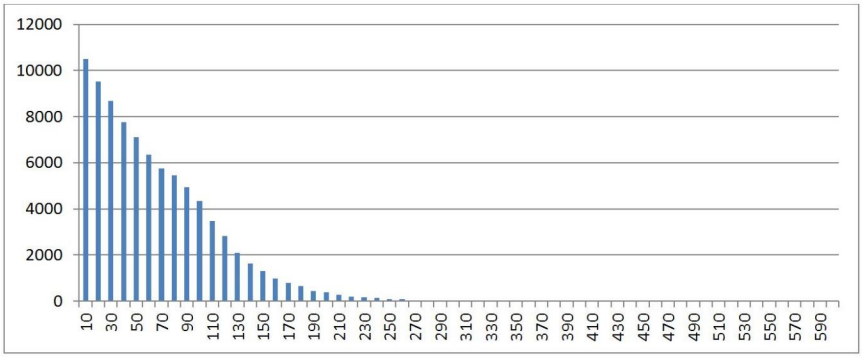

Figure 3. Position of the answer distribution

We observed that the context length mostly lies in the range from 40 to 300 words. Only 0.19% of all contexts are longer than 400 words. Moreover, the answer is typically located within the first 200 words of the context. Closer look at the data showed that there are only a few samples with answer located beyond the 400 words limit (specifically, only 6 samples over 86244 values in train dataset). Therefore, we got an assumption that we can reduce the context length that model operates with from 600 to 400 words without significant loss in prediction quality. Such reduction should not truncate any answers, it might however truncate ~0.19% of the contexts which might affect model outputs on such samples: neural network might have less context for such samples.
In order to prove that reduction of context length from 600 to 400 doesn't lead to reduction in prediction quality and also to measure the performance improvements, we trained the baseline model with context length equal to 400 and compared the results with the baseline. Table 1 shows the difference between 2 experiments.

Experiment	**Dev F1 Score**	**Dev EM Score**	**Running time on 13k iterations**
Context length 600	0.3973	0.2874	5.7 hours
Context length 400	0.3966	0.2886	3.9 hours

Table 1. Experiment results with reduced context length

Results showed that prediction quality of the model with reduced context length is the same as the prediction quality of the original model. We also observe a 31% improvement in training runtime, which is expected since the neural network needs to multiply matrices that are 33% smaller than the original ones. Therefore, we will be using 400 as context length in all the subsequent experiments, this change will allow us to iterate faster with the same quality.
The distribution of the question length appeared to be more even comparing to the context length distribution, we could not reduce its size without affecting the prediction quality.

5 Model improvements

In order to measure how our improvements affect model performance, as the first step we applied the improvements independently of each other. This section treats improvements independently, whereas the next section analyses the model with all the listed improvements combined.

5.1 Bi-directional attention

We implemented bi-directional attention flow (Minjoon Seo, et al., 2017 [1]) and replaced the basic attention layer with it, in comparison to basic attention, bi-directional attention uses a similarity matrix S_{ij} between contextual embeddings representing questions and context, that matrix then is used to generate two attention vectors representing Query to Context (Q2C) and Context to Query (C2Q) attention. The attention vectors are used together

as a blended representation which is the output of the attention layer. Table 2 shows the results of the experiments with bi-directional attention.

Experiment	Dev F1 Score	Dev EM Score
Baseline	0.3973	0.2874
Bi-directional attention	0.4374	0.3166

Table 2. Experiment results with bi-directional attention

5.1.1 Error analysis

Summary of correct and incorrect answers by question words, correct answer are considered as those which have an F1 score higher than 0.50. The results were compiled after analysing 30 random question/answer examples.

Question type	Correct	Incorrect
What	4	11
When	1	1
How	3	1
Which	3	4
Why	N/A	2

Table 3. Question type analysis for bi-directional attention

The word "what" was very frequent on the sampled questions and its number of correct answers are similar to the dev F1 score of the model. The most incorrect answers come from questions starting with "which" which is commonly used for questions that require specific answers or exact matches.

5.2 Multiplicative attention

We wanted to learn more about the impact of using different attention functions. We replaced dot product attention with multiplicative attention (Luong, et. al., 2015a [2]) as defined by: $e_i = s_t W h_i$ the results were similar to the baseline as illustrated in Table 4. In multiplicative attention, the intention of the state matrix W was for its values to be updated as part of the learning process and provide a better attention representation.

Experiment	Dev F1 Score	Dev EM Score
Baseline	0.3973	0.2874
Multiplicative attention	0.3862	0.2800

Table 4. Experiment results with multiplicative attention

The results provided the same performance as using dot product, so it seems that dot product should be used as a default attention mechanism due to less computational complexity. We skip the error analysis section of this improvement since the performance was similar to the baseline.

5.3 Additional features: context-question words match

Baseline model doesn't have any explicit connections between questions and answers in the input data. Both question and answer are being represented as independent word vectors. Nevertheless there are correlations between question and answer tokens that model needs to learn in order to give good predictions of answers.

One of such correlations is the presence of a context word in the question [3]. We analyzed the data in training dataset to see what is the likelihood of a context word to appear in the answer depending on if it appears in the question as well. It turned out that such a likelihood is more than 2 times higher for the context words that appear in the question: **9.5%** of context words that appear in the question also appear in the answer, whereas only **4.7%** of context words that don't appear in the question appear in the answer

There are types of question like "What kind of ..." that typically include the answer word(s) in the question. Such words might be a good anchors for model to pay attention to a part of context that surrounds the word, similarly to how attention layer helps the model to focus only on relevant part of the text. We also noticed that the baseline model performs poorly on the open-ended "Why" questions. We expected that additional attention-like mechanism would help model to focus on relevant parts of context for open-ended questions as well.

Based on that intuition, we added the following features to the model inputs alongside word embeddings:

1. Exact match - "true" if a context word appears in the question and "false" otherwise
2. Lemma match - "true" if lemma of a context word matches lemma of any word in the question and "false" otherwise
3. "Soft" match - additional attention layer that attends context words with every question word and uses weighted sum of attention scores as a feature. This feature should help the model to learn direct similarities between context and question words.

The results of the improvement over the baseline are shown in Table 5. The improvement significantly increases the quality of prediction. Table 6 shows breakdown of scores per question type.

Experiment	**Dev F1 Score**	**Dev EM Score**
Baseline	0.3973	0.2874
Baseline with match features	0.6097	0.4642

Table 5. Experiment results with match features

Question type	**Baseline F1**	**Baseline EM**	**New F1**	**New EM**	**Improvement**
What	0.3664	0.2594	0.5768	0.4288	57.4%
When	0.5467	0.4523	0.7625	0.6452	39.4%
How	0.4571	0.3251	0.6117	0.4439	33.8%
Which	0.4509	0.3432	0.6143	0.4735	36.2%
Why	0.2476	0.0636	0.4698	0.1719	89.7%

Table 6. Experiment results with match features with question type breakdown

As we expected, "What" questions predictions got significant improvements, in addition the model indeed performs much better on the more complicated "Why" questions.

In this particular improvement, we achieved good performance increase with feature engineering. The feature adds valuable signal to the input data (dependencies between context tokens and question tokens) that model doesn't need to learn by itself. Such improvements are great compliments to the end-to-end learning. This shows that whereas deep learning is a powerful tool, it might be still improved by utilizing domain knowledge and allowing the deep learning model to learn on better input data.

5.4 Additional features: word tokens

Paper [3] describes another feature-based improvement we decided to try. For every word in a context and question sentence, we can assign their Part-of-Speech tag, Named Entity Recognition tag and normalized Term frequency,

and then use these additional tokens as input features alongside word embeddings and other features. We implemented the features and got the following improvements over the baseline (Table 7):

Experiment	Dev F1 Score	Dev EM Score
Baseline	0.3973	0.2874
Baseline with token features	0.4113	0.3019

Table 7. Experiment results with word tokens

We observed only minor improvements over the baseline. This feature doesn't perform well because most of the signal that comes from tags is already provided by word vectors. To prove that, we tagged all words in training set with POS tags and then for each unique word vector we counted how many times POS tags belong to one dominant POS category. The ratio of dominant tag count over total tag count shows how much repetitive signal does the POS tagging carry, the closer the ratio to 1, the less signal comes from tagging. Figure 4 shows the ratio dependency on the unique word vector frequency.

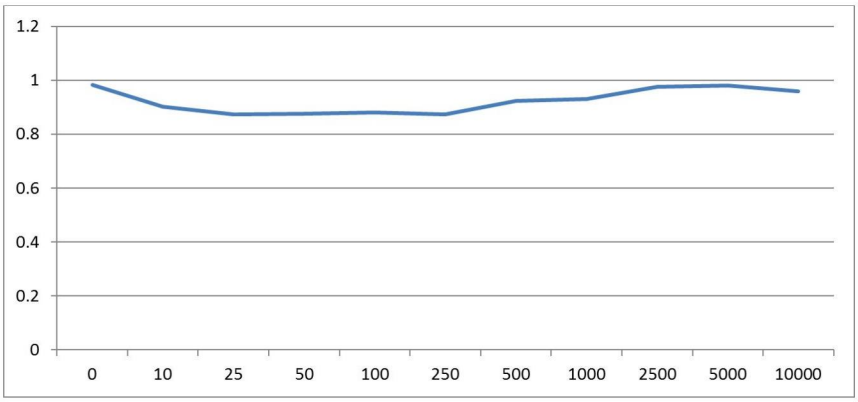

Figure 4. POS tags signal for unique word vector frequency

We observe that the same word is being tagged with the same POS tag most of the times, this is especially true for the most frequent words with frequency 2500+. Therefore tags don't add much input value over word vectors. This explains that F1/EM scores were not significantly improved.

This improvement is a good example of feature that doesn't produce any additional value to the prediction quality. The end-to-end learning approach is capable of learning this feature by itself based on the current inputs that it gets. Our takeaway here is to do research how new features correlate with existing input and if the correlation is high, then the feature engineering will likely not be very effective.

5.5 Adjusting end position based on start position

Besides looking at the aggregated F1/EM scores of the baseline model, we wanted to get more insights about how our start_pos and end_pos predictions look like. Specifically, we wanted to see what is the distribution of the answer length and compare it with the expected distribution from the training set. Figure 5 shows the answer length distribution from the train set. We should expect our model to produce similar distribution.

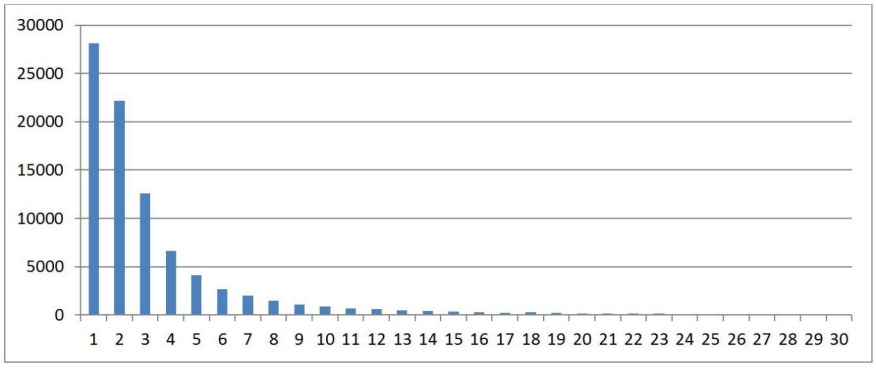

Figure 5. Answer length distribution

We see that most answers are fairly short, 94.6% of all answers have length 10 words or less and 99.7% have length 25 words or less. After having these numbers, we obtained the same distribution of predicted answer length which is shown at Figure 6.

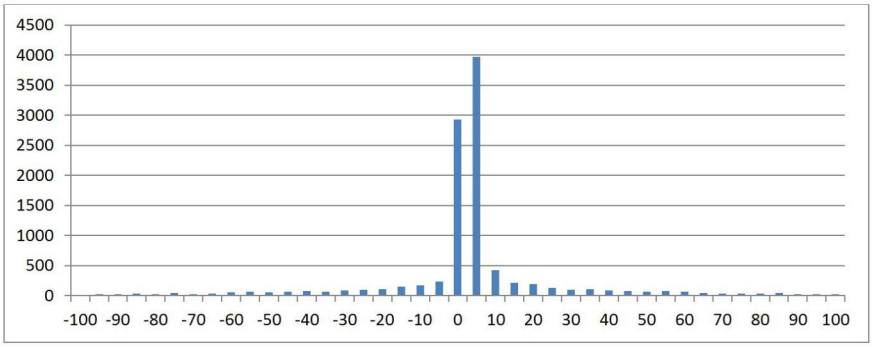

Figure 6. Predicted answer length distribution

This distribution clearly shows 2 flaws that the model has:

1. The model predicts answers with negative length (i.e. with end_pos less than start_pos) in 14.9% of the dev set samples. This prediction doesn't make sense for the problem we're solving
2. The model oftentimes predicts answers that are too long, in 9.3% the answer is longer than 25 words. This contradicts with the distribution we observe in training set where answer is rarely longer than 25 words.

To fix this flaws, we introduced logic of adjusting the end_pos based on the value of start_pos. For a given prediction, after we select start_pos as argmax of start_pos distribution, we mask values of end_pos distribution that are less than start_pos or greater than start_pos + 25. Therefore, the argmax of end_pos distribution only selects the max value from the [start_pos, start_pos+25] range. Table 8 shows the improvement we achieved by applying this adjustment

Experiment	**Dev F1 Score**	**Dev EM Score**
Baseline	0.3973	0.2874
Baseline with adjusted end_pos	0.4686	0.3384

Table 8. Experiment results with adjusted end_pos

This simple adjustment leaded to increase in the prediction quality. The 17.9% increase in F1 metric roughly corresponds to the fact that we re-predict 24.2% of the samples with originally incorrect predictions

6 Results and analysis of the ensemble model

We achieved F1 score 68.2 and EM 56.3 on dev set, and F1 score 68.5 and EM 56.9 on test set. Such results indicate significant improvement over the baseline model that had F1 score 39.7 and EM 28.7 on dev set. It is still below the state-of-the art models performance which have F1 scores around 80. We took a closer look at the specific examples on which our model doesn't perform well in order to figure how the model can be improved further.

6.1 Error analysis

We randomly selected 50 incorrect questions with a zero EM score and categorized the errors into the following categories: 53% wrong boundaries, 30% syntactic complication and ambiguities, for an instance, model answer being "10" and true answer "10 counties" , 13% paraphrase problems , 4% external knowledge required. Here are a few samples from the most frequent categories.

6.1.1 Wrong boundaries (53%):

- Sample context: "the harvard business school and many of the university 's athletics facilities , including harvard stadium , are located on a _358-acre_ (145 ha) campus opposite the cambridge campus in allston . the john w. weeks bridge is a pedestrian bridge over the charles river connecting both campuses ."
- Sample question: "what is the name of the bridge that joins parts of the campus of the charles river ?"
- Predicted answer: "cambridge campus in allston . the john w. weeks bridge"
- True answer: "the john w. weeks bridge"

The model correctly identified the part of the context with the answers but failed to extract the exact location of the answer. More comprehensive attention techniques like self-attention should help with this type of errors.

6.1.2 Syntactic complication and ambiguities (30%):

- Sample context: " to counter the french military presence in ohio , in october 1753 dinwiddie ordered the 21-year-old major george washington (whose brother was another ohio company investor) of the virginia regiment to warn the french to leave virginia territory ."
- Sample question: "who did dinwiddie order to address french in virginia territory ?"
- Predicted answer: "george washington"
- True answer: "major george washington"

The model correctly identified key words of the answer but failed to include a modifying words into the answer. This might be an indication of inconsistencies in the training set, some of the samples might have modifying words whereas others might not.

7 Conclusion

For the final project assignment, we implemented several improvements over the baseline question answering model. We added bi-directional layer, match-based features, token-based features as well as made improvements to end pos prediction. We improved the F1 score up to 68.5 and EM score up to 56.9 on test set. We analyzed the results of our experiments, provided error analysis and noticed possible further improvements.

During the assignment, we learned how to build deep learning systems to solve NLP problems, how to analyze their performance. We learned advanced attention techniques and the tradeoffs of feature engineering vs. end-to-end learning.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi, Bidirectional attention flow for machine comprehension, ICLR, 2017.

[2] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention based neural machine translation. In EMNLP.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.