Question Answering on SQuAD

Evan Darke
Department of Computer Science
Stanford University
Stanford, CA 94305
edarke@stanford.edu

Jake Smola
Department of Computer Science
Stanford University
Stanford, CA 94305
jsmola@stanford.edu

Abstract

Open domain question answering is a complex and widely applicable task in NLP where a machine attempts to answer a query by selecting a relevant phrase from a given context. This paper will examine techniques for question answering on the Stanford Question Answer Dataset (SQuAD) by combining Bidirectional Attention Flow (BiDAF) with feature engineering and propose an improved span selection method.

1 Introduction

An exciting problem in the field of Natural Langauge Processing concerns the task of reading comprehension, commonly implemented in the form of question answering. A simple yet well-trained model can take a paragraph (context) and question as input, and output indices corresponding to the start and end points of the answer as found in the context.

The Stanford Question Answering Dataset (SQuAD) is a public source of data comprised of roughly 100,000 contexts, questions, and answers, which facilitate the training and evaluation of NLP models in reading comprehension. Table 7 shows an example of a context, question, and answer tuple. Figures 1 and 2 illustrate the distribution of question types and the mean question length by type respectively.

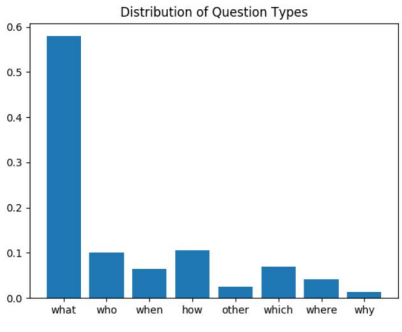

Figure 1: Categorization of question types in training set

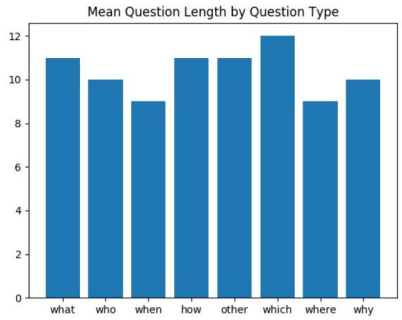

Figure 2: Question length by type in training set

The "SQuAD Challenge" is a public competition which rewards machine learning models based on their performance on two criteria: [4]

- F1 Score: Computed as the harmonic mean of the precision and recall scores.

- Exact Match (EM) Score: Yields 1 if the answer prediction matches the ground truth exactly, and 0 otherwise.

We start with a baseline model consisting of the system depicted in Figure 3.

In this paper, we describe our own models and experiments in seeking to maximize the aforementioned metrics. Our investigation explored the following model features as modular improvements to the given baseline:

- Character-level embedding layer
  - Convolution & max pool layers
- Bidirectional attention flow (Bidaf) layer
- Exact match feature
- Span representation feature

Our comprehensive model appears in Figure 4.

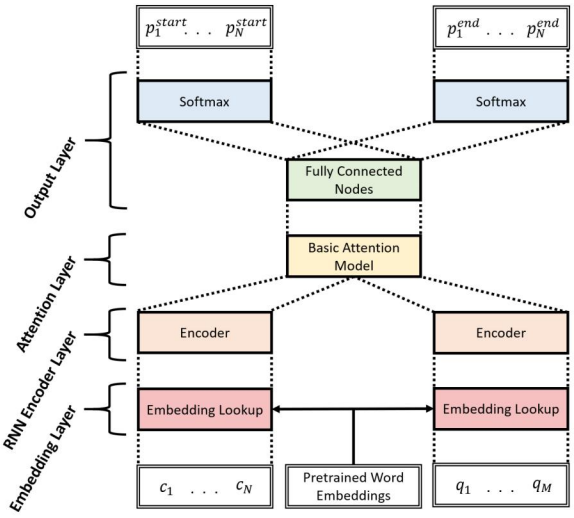

Figure 3: Baseline model.

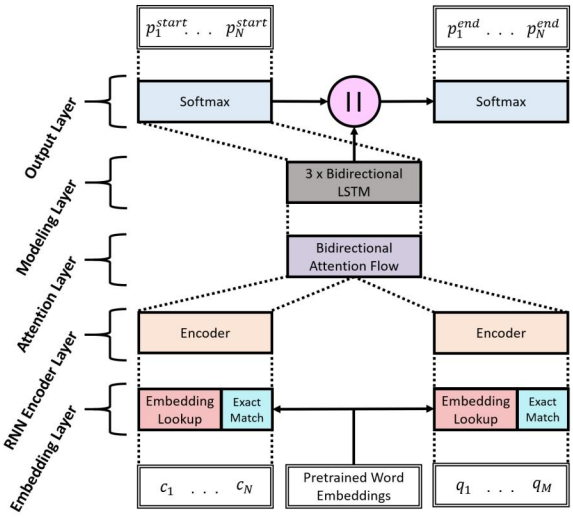

Figure 4: Full model.

2 Previous Work

Much successful work on question answering has resulted from applying attention mechanisms to recurrent neural networks. Attention mechanisms help the network identify which words in the context are most relevant to which words in the query. In particular BiDAF, which combines context-to-query and query-to-context attention, achieved state of the art results on multiple question answering datasets [5].

Although word embeddings alone are sufficient to obtain good performance on this task, [3] showed that networks can benefit significantly by supplementing word embeddings with hand-engineered features. The authors define $f_{exact\_match}(w_i)$ for context words as a binary feature which has value 1 if the word w_i also appears in the question.

Both papers compute start and end pointers that correspond to the index of the first and last word of the answer in the context. This is done by a softmax layer for each pointer with number of classes equal to length of the context.

Another approach involves the use of passage-aligned question representations, which are computed using query-to-context attention [2]. These representations can enhance question-answering done using passage-independent representations alone as they allow the model to "align" the question with the context to focus on the region of the context most likely containing the answer.

Other approaches include leveraging character-level embeddings. The model in [5] tokenizes question and context inputs at the character level prior to computing character-level embeddings. These embeddings are later passed through a convolution layer and a max-pooling layer to assemble the character-embeddings into word-embeddings which are then concatenated with pre-trained word embeddings.

3 Approach

3.1 Baseline model

Our baseline model uses 100-dimensional pre-trained GloVe embeddings to encode each batch of contexts (maximum N words) and questions (maximum M words). The embeddings are fed into a bidirectional RNN layer which produces hidden states associated with the embeddings in each cohort. We then compute basic dot product attention with the context hidden states attending to the question hidden states. The attention distribution is concatenated with the context hidden states and is fed into a dense layer with a ReLU activation function. This output is then fed into two softmax layers which compute the probability distribution over start and end positions, denoted p^{start} and p^{end}. The span is selected by taking the argmax of p^{start} and p^{end}.

This model requires fixed size context length N and question length M. M and N were initially chosen to be 600 and 30 respectively. Inputs are padded or truncated as necessary.

3.2 Improvements

As in [3], we append to every word embedding the binary feature f_{exact_match} which has value 1 if the word appears in the question and 0 otherwise. We define f_{exact_match} as 1 for all question word embeddings to ensure that context and question embeddings have equal dimension. Whereas [3] considers word to match if their original forms, their lemma forms, or lowercase forms match, we only consider words to match if they have the same word ID, for computational efficiency.

Additionally, we investigated using a character-level CNN to extract features from each word and append these to their word vector representation. We trained a character embedding matrix representing 79 alphanumeric characters and symbols commonly found in text (including mathematical operators, punctuation, parentheses, braces, and units) with an embedding size of 20. For each word, we computed a char embedding matrix for the word and ran 100 1-dimensional convolutional filters with kernel size 5 to extract character level features for each word. We then take an row-wise max over these features and append the resulting vector to our word embeddings. These character level features help distinguish tokens for which there are no word embeddings.

We also replace our basic dot product attention with Bidirection Attention Flow (BiDAF) as described in [5]. For context hidden states c_i and question hidden states q_i, we compute the following for a learned parameter $w \in R^{6h}$ [1]

$$
\begin{aligned}
S_{i,j} &= w^T[c_i; q_j; c_i \circ q_j] \\
\alpha^i &= softmax(S_{i,:}) \\
a_i &= \sum_{j=1}^{M} \alpha^i_j q_j \\
m_i &= \max_j S_{i,j} \\
\beta &= softmax(m) \\
c' &= \sum_{i=1}^{N} \beta_i c_i \\
output_i &= [c_i; a_i; c_i \circ a_i; c_i \circ c']
\end{aligned}
$$

We also replace the dense ReLU layer with a stack of 3 BiLSTMs. We will refer to this module as the modeling layer of the network.

Additionally, we append the softmax output of the start distribution to the input for the output layer for p^{end}. This way, the network is able to predict the end index based on its prediction of the start.

We optimize the computational efficiency of the model by plotting the distribution of question and context lengths in figures 5 and 6. We chose to decrease our context length N to 400 because this handles nearly all examples without truncation. We allow M to remain 30, although this may be more conservative than necessary. This optimization speeds up iterations by a factor of $\approx \frac{1}{3}$.

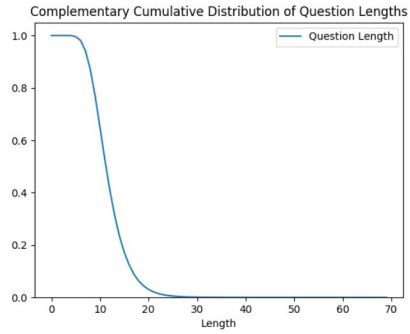

Figure 5: Plot of $P(Question\ length > X)$

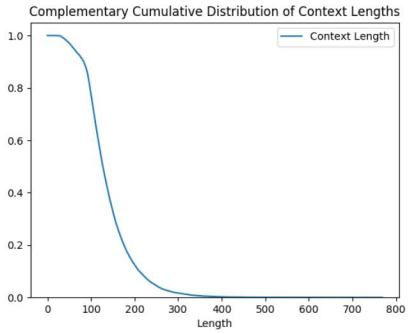

Figure 6: Plot of $P(Context\ length > X)$

Finally, we explore alternative span selection techniques described in 3.3

3.3 Improved span selection

The baseline model selects a span that maximizes $p_i^{start} p_j^{end}$. In [3], the selected span also maximizes the above expression with the additional constraint that $0 <= i <= j <= i + 15$. However, these selection methods can do poorly if the network identifies two candidate answers, in which case taking the argmax of this product may not correspond to selecting either candidate answer.

To address this shortcoming, we propose a new span selection approach that incorporates a prior belief over the expected length of the correct answer. Our span selection algorithm is parameterized by an upper triangular matrix $P \in R^{N \times N}$ where $P_{i,j}$ represents the probability of a correct answer having length $j - i + 1$. At test time, we select our span as follows:

$$span = \underset{i,j}{\operatorname{argmax}}\, p_i^{start} p_j^{end} P_{i,j} = \underset{i,j}{\operatorname{argmax}}(p^{start} p^{end^T} \circ P)_{i,j}$$

Compared to the baseline model which selects the optimal span in $O(N)$ time given the start and end pointer distributions, our method requires $O(N^2)$ time.

Span selection in [3] is a special case of this model where the underlying distribution is assumed to be uniform over positive integers less than 16. In this paper, we propose two alternative ways to define P. First, we measure the empirical distribution of answer lengths from the training set shown in figures 7 and 8 and initialize P accordingly with Laplace smoothing. We will refer to this P as our empirical prior.

Secondly, we note that the answer length is not independent of the question content. For example, figure 8 shows that answers to 'why' questions are significantly longer than answers to 'who' questions, which are often just a first and last name. Therefore, we also have the model learn to predict the length of the answer given the question. We do this by taking the elementwise max of the question embeddings over each timestep and feeding the resulting vector into a softmax layer that is trained to predict the answer length. The output of this softmax is used to construct an independent P matrix for each example during span selection. We will refer to this P as our learned prior.

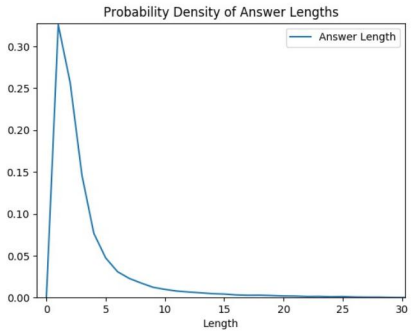

Figure 7

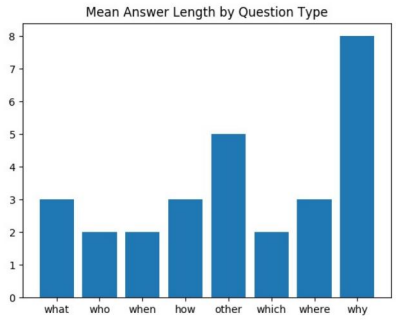

Figure 8

4 Experiments

Before arriving at a comprehensive model, we conducted a variety of experiments to tune and improve the baseline. Initially, we investigated the improvements detailed in section 3.2. Our initial analysis yielded an improved training baseline upon which we could conduct further experiments. This improved baseline consisted of an exact match feature, bidirectional attention flow, a convolution layer, 200 hidden units, and 15% dropout.

4.1 Comparative Analysis

In table 1, we detail a comparative analysis which covers incremental changes to the static, improved baseline. Given the training and development set statistics collected during this experiment and our initial investigation, we knew the improved baseline tended to overfit to the training data.

This precipitated our investigation of some regularization methods which could potentially improve our results. Table 2 details our experiments with L2-regularization and dropout.

Table 1: Comparative Analysis after 14000 iterations for improved baseline consisting of: Exact Match feature, BiDAF, 2 BiLSTMs, Conv. layer, 200 hidden units

Model	Dev F1	Dev EM
Improved baseline	67.53	52.79
150 hidden units	**+0.12 %**	-0.86 %
100 hidden units	**+0.58 %**	**+0.26 %**
50 hidden units	-2.06 %	-3.12 %
Ignore non-alpha-numeric characters	**+0.24 %**	-0.29 %
Replace Convolution with Dense Layer	-0.32 %	-0.44 %
Add Char-Level Embeddings w/ CNN	-2.91 %	-4.63 %

Table 2: Regularization Comparative Analysis after 14000 iterations for baseline consisting of: Exact Match feature, BiDAF, 2 BiLSTMs, Conv. layer, 200 hidden units

Model	Train F1	Train EM	Dev F1	Dev EM
Improved baseline (15% dropout)	91.26	81.8	67.53	52.79
L2-Regularization ($\beta = 0.01$)	54.65	37.00	52.16	35.38
L2-Regularization ($\beta = 0.01$) 10% dropout	56.27	38.50	53.01	36.73
L2-Regularization ($\beta = 0.01$) 5% dropout	56.70	40.00	54.01	37.40
L2-Regularization ($\beta = 0.001$) 10% dropout	73.49	58.80	65.93	50.56
L2-Regularization ($\beta = 0.0001$) 10% dropout	89.87	78.5	67.41	**53.37**

Additional investigation concerned a variety of other hyperparameters including the number of BiLSTM's, embedding size, and others. These analyses showed that including more LSTM's tended to improve accuracy while slowing training.

4.2 Ablative Analysis

Continued investigation and the aforementioned experiments, combined with periodic official evaluation in our CodaLab environment, facilitated our arrival at a final model. Table 3 shows the effect of removing features one at a time from this model until we are left with the LSTM baseline in terms of the cost to the F1 score on the development set. Independent indices refers to removing p^{start} as an input feature for the p^{end} calculation. This shows that the primary benefit of our improvements come from the 3 BiLSTM layers that compose the modeling layer.

4.3 Span selection

Table 4 shows the results of four span selection methods applied to the same trained network at test time. This shows that incorporating a prior distribution gives better results, especially on the exact match metric, than simply taking the argmax as in the baseline or constraining answers length to be bound by some hyperparameter as in DrQA. Surprisingly, the learned prior did worse than using a uniform prior. We believe this is the result of overfitting due the to rich feature space of the question embedding. Furthermore, we found that applying the empirical prior to the training set decreases the F1 and exact match score from 95% and 88% to 92% and 82% respectively. We believe incorporating a prior belief over the answer length has a regularizing effect on the model because the end pointer output layer is implicitly learning to predict answer lengths from the contents of the question and context.

Table 3: Ablative Analysis

Model	Dev F1	Cost
Full Model	72.31	- -
No empirical prior	70.81	1.50
Independent indices	70.06	0.75
No f_{exact_match}	67.85	2.21
No modeling layer	45.89	21.96
No BiDAF	42.05	3.84

Table 4: Performance of Span Selection Methods

Model	Dev F1	Dev EM
Baseline	70.81	64.07
DrQA (bounded uniform prior)	71.64	65.06
Empirical Prior	**72.31**	**67.04**
Learned Prior	71.59	64.42

5 Analysis

Tables 5 and 6 compare our baseline and final model performance. Based on our ablative analysis in Table 3, the modeling layer composed of three bidirectional LSTM's had the greatest impact on our overall F1 score. This is expected since the modeling layer comprises the main sequence model for associating query and context terms in a manner successful in various language comprehension tasks. Training these BiLSTM's allowed our model to semantically interpret each context and question and consequently answer questions with improved accuracy.

Table 5: Baseline Performance

Model	Dev F1	Dev EM
Baseline (GRU)	40.58	29.42
Baseline (LSTM)	**42.05**	**30.62**

Table 6: Final Model Performance

Model	Test F1	Test EM
Final Single Model	75.11	66.36

The presence of a bidirectional attention flow layer as opposed to basic dot-product attention had the second highest impact on our overall performance. Although the impact was marginal compared to our modeling layer, the BiDAF layer computes the similarities of context terms relative to query terms and vice-versa. Compared to the baseline's basic attention model, which only computes similarity of context terms relative to query terms, BiDAF is able to attend to relevant embeddings in both directions and hence retain key information from each question that may be otherwise lost.

The third most influential feature of our model was the Exact Match feature. It is reasonable to expect the answer to a question to contain terms present within the question itself and so it comes as no surprise that prioritizing these terms improves our F1 score (which is less strict than EM). Of

course, it is trivial to come up with questions that share words with the context but not the answer. This possibility likely explains why this feature only offered marginal gains.

Figures 9 and 10 show the performance of our model on a development set with official evaluation. The model performs similarly on different types of questions, but performance degrades significantly on answers longer than 4 tokens. This is may be due in part to our selection method being adverse to selecting long answers.

Table 7 below shows examples of good and poor predictions on the development set. In the first example, the model likely fails to predict _ogedei_ because it appears as an unknown token. It does, however, successfully choose a name in close proximity to 'grandson'. Additional input features beyond word embeddings may help address such cases. The second example demonstrates that our model is still capable of correctly selecting answers as long as 13 words, even though such answers are extremely unlikely according to our empirical prior.

Table 7: Samples from dataset with prediction from model with empirical prior

Context: instability troubled the early years of kublai khan 's reign . _ogedei_ 's grandson kaidu refused to submit to kublai and threatened the western frontier of kublai 's domain . the hostile but weakened song dynasty remained an obstacle in the south . kublai secured the northeast border in 1259 by installing the hostage prince _wonjong_ as the ruler of korea , making it a mongol tributary state . kublai was also threatened by domestic unrest . li tan , the son-in-law of a powerful official , instigated a revolt against mongol rule in 1262 . after successfully suppressing the revolt , kublai curbed the influence of the han chinese advisers in his court . he feared that his dependence on chinese officials left him vulnerable to future revolts and defections to the song .	**Query:** who was kaidu 's grandfather ? **Ground Truth:** ogedei **Predicted:** kublai khan
Context: european union law is applied by the courts of member states and the court of justice of the european union . where the laws of member states provide for lesser rights european union law can be enforced by the courts of member states . in case of european union law which should have been transposed into the laws of member states , such as directives , the european commission can take proceedings against the member state under the treaty on the functioning of the european union . the european court of justice is the highest court able to interpret european union law . supplementary sources of european union law include case law by the court of justice , international law and general principles of european union law .	**Query:** who applies european union law ? **Ground Truth:** courts of member states and the court of justice of the european union **Predicted:** courts of member states and the court of justice of the european union

Our final hyperparameters are displayed in Table 8.

6 Conclusion

In this paper, we investigated question answering models on SQuAD. We showed that incorporating a prior belief into span selection increases F1 and EM scores on the development set without introducing any additional training cost. One drawback of our model is that we do not explicitly handle out-of-vocabulary words. This could be addressed by including a Char-Level CNN and tuning hyperparameters to overcome the overfitting we saw with our initial investigation, displayed in table 2. Additionally, we could modify our f_{exact_match} feature to consider words as strings, rather than

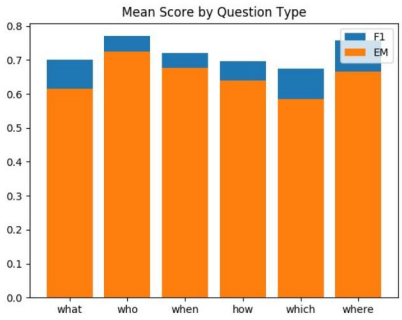

Figure 9

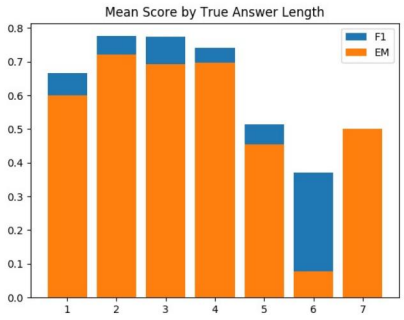

Figure 10

Table 8: Final Hyperparameters

Parameter	Value
Learning Rate	.001 annealed to .00025
Batch Size	64
Dropout	.20
Context Length	400
Question Length	30
Embedding Size	200
Max Gradient Norm	5.0

just IDs. This would allow the model to recognize when the same unknown word appears in both the question and context.

References

[1] Cs 224n default final project: Question answering. `http://web.stanford.edu/class/cs224n/default_project/default_project_v2.pdf`. Accessed: 2018-03-18.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051, 2017.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.