Towards an integrated question-answering model

Fangzhou Liu
fzliu96@stanford.edu

Abstract

This paper builds on existing work on the Stanford Question-Answering Dataset (SQuAD), constructing various models that aim to select an answer span from a longer context paragraph in response to a factual question. I aim to integrate various high-performing SQuAD models such as R-Net and BiDAF by experimenting with different combinations of word embedding representations, attention layers as well as output layers. The most successful model on SQuAD is a combination of the bi-directional context-question attention layer in BiDAF with hybrid word-character representations combined using fine-grained gating, rather than concatenation.

1 Introduction

In this paper, I construct and compare various models that select an answer span from a longer context paragraph in response to a factual question, trained and evaluated on the SQuAD dataset. My aim is twofold: First, to test the effectiveness of hybrid word- and character-level embedding representations compared to pure word presentations, as well as methods of combining these representations. Second, to compare the effectiveness of various output layer mechanisms in combination with the base bi-directional attention (context-to-question and question-to-context) borrowed from the BiDAF model implemented by Seo et al 2016.

2 Data

First, a key shortcoming of the baseline model is that it predicts the end of the answer span independently of the start. Several answer are predicted wrongly because the end token chosen occurs before the start token, so that no answer span is chosen in effect. Thus conditioning the end prediction on the start prediction is a priority for this model.

Second, the baseline model saw severe overfitting even though dropout was applied at the basic attention layer. Dev F1 and train F1 scores diverge so that after 15,000 iterations, the best F1 dev score is 0.4 while the best F1 train score is 0.75.

This suggests that hyperparameter tuning is required on at least two fronts: first, increasing dropout, and second, generally identifying strategies to reduce parameter dimensions even as model complexity grows with additional layers. There is a trade-off between adding new attention layers and the complexity of the hidden representation of each word: As new layers are added, the most obvious and effective way of restricting model complexity is to reduce the size of the hidden layer since it is used throughout the model.

Third, on setting basic hyperparameters including answer length, context length and question length: Answers clearly tend to be brief, with 95.2% of the 10,000 training answers sampled being 10 tokens or shorter in length. The length of contexts and questions of 10,000 training examples also show that a maximum question length of 30 and a maximum context length of 400 or 500 are appropriate values.

3 Previous work

In designing my models, I draw on three chief resources: Bi-directional Attention Flow (Seo et al 2016), R-Net (Microsoft Research Asia 2016) and fine-grained gating applied to mixed word-character representations (Yang et al 2017). The final product is based most heavily on the BiDAF model, using its context-to-query and query-to-context attention mechanism to allow each context token to attend to each question token.

Each experiment I conducted included the bi-directional attention representation; beyond that, I compare two main features of the model: the choice of word embedding method and the output layer. Following Yang et al 2017, I use a weight vector to add the word and character-level embeddings of each word to obtain the final embedding of the word; I then compare this with a representation that concatenates word- and character-level embeddings. [1]

For the output layer, I compare a baseline that calculates start- and end- probability distributions independently, the original BiDAF model's output layer as well as the answer pointer from R-Net. Each modeling decision is described more in detail in the following section.

To complement the answer pointer, I also experimented with a self-attention layer based on R-Net; however, this saw severe overfitting (training and validation F1 scores of 0.8 and 0.4 respectively), likely due to the model complexity of a BiDAF attention representation combined with self-attention. To prioritize attending to the question, I chose not to pursue this model further.

4 Model

The model consists of three primary layers: (1) word- and character-level embeddings, (2) the bi-directional RNN layer, which incorporates bi-directional attention as well as self-attention at each step, (3) an output layer that generates start and end probability distributions. Each timestep of the core bidirectional RNN model is comprised of a GRU cell, which performs well and conserves memory compared to a LSTM cell.

4.1.1 Embeddings

Word-level embeddings are obtained using pre-trained GloVe vectors as in (1a) while character-level embeddings are obtained using a Character CNN model as in (1b), described in greater detail below.

$$\text{GloVeEmb}(\boldsymbol{w}_i) \in \mathbb{R}^{Dw} \quad (1a) \qquad \text{CharCNN}(\boldsymbol{w}_i) \in \mathbb{R}^{Dc} \quad (1b)$$

I use a learnable embedding matrix CharEmb to encode each character as a vector $\boldsymbol{v}_{ce} \in \mathbb{R}^{Dce}$. I concatenate the vectors $\boldsymbol{v}_{ce}$ to obtain $\boldsymbol{w}_{i,\,ce} = [\boldsymbol{v}^1_{ce} ; \dots ; \boldsymbol{v}^L_{ce}]$, where L = 30 is the (padded) maximum number of characters in a word. Each word is fed into a 1D convolutional layer with D_c total filters of size k each, followed by a MaxPool layer using a RelU non-linearity:

$$\boldsymbol{w}_{i,\,conv} = \text{Conv1D}(\boldsymbol{w}_{i,\,ce}) \;.$$

$$\text{CharCNN}(\boldsymbol{w}_i) = \text{MaxPool}(\text{RelU}(\boldsymbol{w}_{i,\, conv})) \in \mathbb{R}^{Dc}$$

4.1.2 Fine-grained gating

Existing models that apply Character CNNs to extractive question-answering have used concatenation to combine character- and word-level representation.

This model compares concatenation with fine-grained gating, assuming in both cases that $D_w = D_c$, where $_{c}$ is both the dimension of the output character CNN encoding and the number of filters applied in convolution. Following Yang et al 2017, fine-grained gating was defined based on the following equations, using a slightly modified $\boldsymbol{v}$ vector: [1]

$$\boldsymbol{g}_i = \sigma(\mathrm{W}_h \boldsymbol{v}_i + \boldsymbol{b}_h) \in \mathbb{R}^{Dh}$$

$$\boldsymbol{l}_i = \boldsymbol{g}_i \circ \text{CharCNN}(\boldsymbol{w}_i) + (1 - \boldsymbol{g}_i) \circ \text{CharCNN}(\boldsymbol{w}_i) \in \mathbb{R}^{Dh}$$

Here, $\boldsymbol{v}_i \in \mathbb{R}^{Dv}$, $\mathrm{W}_h \in \mathbb{R}^{Dh \times Dv}$ and $\boldsymbol{g}_i$, $\boldsymbol{b}_i \in \mathbb{R}^{Dh}$; W_h and $\boldsymbol{b}_h$ are learnable parameters. The symbol '$\circ$' denotes element-wise multiplication. Note $D_h = D_c = D_w$ in this model; i.e. this particular approach restricts hyperparameter tuning since the character embedding, word embedding and hidden layer must have the same dimensionality.

$\boldsymbol{v}_i$ is the concatenation of the following: the word-level representation GloVeEmb($\boldsymbol{w}_i$), a one-hot encoding of the part of speech tag for $\boldsymbol{w}_i$, and an indicator value **1**{GloVeEmb($\boldsymbol{w}_i$) = '<UNK>'} (i.e. the out-of-vocabulary GloVe token.)

The out-of-vocabulary indicator token directly represents one of the primary use cases of the character-level representation. The part-of-speech tag enables us to encode the distinction between lexical parts of speech such as nouns and verbs, which see greater morphological variation (e.g. *mouse* vs. *mice* and *run* vs. *ran*), compared to functional parts of speech such as conjunctions and cardinal numbers (e.g. *or* and *three*, which both have only one form in English.) The lexical-functional distinction is observed cross-linguistically, even though the mapping from parts of speech to lexical vs. functional categories may differ. Thus these additional features do not restrict the generalizability of the model across languages.

After the hidden representations are obtained for both directions of the RNN, I concatenate the forward and backward hidden state for each context or question word to obtain hidden representations as follows:

context: $\boldsymbol{c}_1, \ldots \boldsymbol{c}_N \in \mathbb{R}^{2Dh}$

query: $\boldsymbol{q}_1, \ldots \boldsymbol{q}_M \in \mathbb{R}^{2Dh}$

4.2 Bi-directional attention flow

Following Seo et al 2016, I implement a bidirectional attention model that calculates both context-to-query (C2Q) and query-to-context (Q2C) attention.[2] Q2C captures the semantic similarity or 'relevance' of certain context words to certain query words and hence are useful in answering the question; C2Q allows each context word to attend to certain query words that are most relevant. For instance, the correct answer span to the query below includes one token from the query, *etude*; it would be ideal to derive a high attention score both for C2Q and Q2C.

QUESTION: **in which etude of neumes rythmiques do the primes 41 , 43 , 47 and 53 appear in ?**
ANSWER: **the third étude**

C2Q attention

First, I obtain a matrix $S \in \mathbb{R}^{N \times M}$ by defining:

$$S_{ij} = \boldsymbol{w}^{\mathbf{T}}_{sim} [\boldsymbol{c}_i ; \boldsymbol{q}_j ; \boldsymbol{c}_i \circ \boldsymbol{q}_j] \qquad [2]$$

where $\boldsymbol{w}^{\mathbf{T}}_{sim} \in \mathbb{R}^{6Dh}$ is a learned parameter and the notation above denotes the concatenation of the three vectors. S can be interpreted as a similarity or relevance matrix between every context token represented $\boldsymbol{c}_i$ and every question token represented by $\boldsymbol{q}_j$. Then, I obtain the row-wise softmax of S to obtain the attention distribution $\boldsymbol{\alpha}_i$ for each $i = \{1, \ldots, N\}$, and corresponding attention output $\boldsymbol{a}_i$:

$$\boldsymbol{\alpha}_i = \mathrm{softmax}(S_{i,:}) \in \mathbb{R}^{M}$$
$$\boldsymbol{a}_i = \sum_j \boldsymbol{\alpha}^j_i \, \boldsymbol{q}_j \in \mathbb{R}^{2Dh} \qquad [2]$$

Q2C attention

Second, re-using the similarity matrix S, I take the row-wise maximum of S and take the softmax over the vector $\boldsymbol{m} \in \mathbb{R}^{N}$, thus obtaining an attention distribution $\boldsymbol{\beta}$ over all context states that gives an attention output by summing over all context states $\boldsymbol{c}_i$, $i = \{1, \ldots, N\}$.

$$\boldsymbol{m}_i = \max_j S_{ij} \in \mathbb{R} \quad \forall i \in \{1, \ldots, N\}$$
$$\boldsymbol{\beta} = \mathrm{softmax}(\boldsymbol{m}) \in \mathbb{R}^{N}$$
$$\boldsymbol{c}' = \sum_i \boldsymbol{\beta}_i \, \boldsymbol{c}_i \in \mathbb{R}^{2Dh} \qquad [2]$$

This layer returns the output as below, combining Q2C and C2Q attention with the hidden representation of the context token itself through elementwise multiplication.

$$\check{\boldsymbol{c}}_i = [\boldsymbol{c}_i ; \boldsymbol{a}_i ; \boldsymbol{c}_i \circ \boldsymbol{a}_i; \boldsymbol{c}_i \circ \boldsymbol{c}'] \in \mathbb{R}^{8Dh} \qquad [2]$$

Between the bi-directional attention layer and the answer layer, the bi-directional attention outputs $\check{\boldsymbol{c}}_i$ are fed into a bidirectional RNN. and the original vectors $\boldsymbol{v}_i$ are concatenated and encoded again as a bidirectional RNN (using GRU cells to avoid vanishing gradients).

$$\{\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \} = \mathrm{BiGRU}(\{\check{\boldsymbol{c}}_1, \ldots, \check{\boldsymbol{c}}_N\}) \in \mathbb{R}^{16Dh} \qquad [2]$$

Here, the RNN allows information from previous context tokens, represented as RNN hidden states, to propagate down to future context tokens, capturing some information about the relevance of previous tokens to future ones without complex self-attention mechanisms such as the one in R-Net.

4.3 Output layer

To generate start- and end-index probability distributions, I experimented with two output layers in addition to a baseline method: an answer pointer, and a layer based on the BiDAF model.

The baseline simply takes the softmax over the output $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ from the BiDAF layer two separate times, once to generate the start distribution and once to generate the end distribution:

$$\boldsymbol{p}^{i}_{start} = \mathrm{softmax}(\mathbf{v}^{\mathrm{T}}_{start}\, \boldsymbol{h}_i)$$
$$\boldsymbol{p}^{i}_{end} = \mathrm{softmax}(\mathbf{v}^{\mathrm{T}}_{end}\, \boldsymbol{h}_i)$$

4.3.1 Answer pointer

Thus this final encoding of the context tokens at each time-step captures the token's own semantic and syntactic features, its query-relevance as well as the relevance of other context tokens to its meaning.

At this point, the answer pointer is simply the application of two additive attention-like layers. Below, $\{\boldsymbol{a}_1^{\mathrm{start}}, \ldots ,\boldsymbol{a}_N^{\mathrm{start}}\}$ is the start probability distribution over all the context tokens.

$$\boldsymbol{s}_j^{\mathrm{start}} = \boldsymbol{u}^{\mathrm{T}} \tanh(\mathrm{W}^{h}\boldsymbol{h}_j + \mathrm{W}^{r}\boldsymbol{r}) \in \mathrm{R}$$
$$\boldsymbol{a}^{\mathrm{start}} = \mathrm{softmax}(\boldsymbol{s}^{\mathrm{start}}) \in \mathrm{R}^{N} \qquad [3]$$

$\boldsymbol{r} \in \mathbb{R}^{2Dh}$ is a weighted representation over the question hidden states, and is used as input to $\boldsymbol{s}^{\mathrm{start}}$: thus the start pointer can be seen as an attention distribution of the new context hidden states $\boldsymbol{h}_j$ over the combined question states, representing p(*start* | **Q**).

$$\boldsymbol{s}_j^{q} = \boldsymbol{u}^{\mathrm{T}} \tanh(\mathrm{W}^{q}\boldsymbol{q}_j + \boldsymbol{b}^{q}) \in \mathrm{R}$$
$$\boldsymbol{a}^{q} = \mathrm{softmax}(\boldsymbol{s}^{q}) \in \mathrm{R}^{M}$$
$$\boldsymbol{r} = \textstyle\sum_j \boldsymbol{a}_j^{q}\, \boldsymbol{q}_j \in \mathbb{R}^{2Dh} \qquad [3]$$

Finally, I use the attention distribution $\boldsymbol{a}_j^{\mathrm{start}}$ to obtain output $\boldsymbol{r}_{out}$, which replaces $\boldsymbol{r}$ as input to the end token distribution $\{\boldsymbol{a}_1^{\mathrm{end}}, \ldots , \boldsymbol{a}_N^{\mathrm{end}}\}$. This, in turn, allows the context hidden states to attend to the start token attention output, establishing the dependency of the choice of end token on start token, representing p(*end* | *start*).

$$\boldsymbol{r}_{out} = \textstyle\sum_j \boldsymbol{a}_j^{start}\, \boldsymbol{h}_j \in \mathbb{R}^{2Dh}$$
$$\boldsymbol{s}_j^{end} = \boldsymbol{u}^{\mathrm{T}} \tanh(\mathrm{W}^{h}\boldsymbol{h}_j + \mathrm{W}^{r}\boldsymbol{r}_{out}) \in \mathrm{R}$$
$$\boldsymbol{a}^{\mathrm{end}} = \mathrm{softmax}(\boldsymbol{s}^{\mathrm{end}}) \in \mathrm{R}^{N} \qquad [3]$$

Here, $\mathrm{W}^{h}, \mathrm{W}^{r} \in \mathbb{R}^{2Dh \times 2Dh}$ by necessity to maintain consistent dimensions with the question hidden states; $\mathrm{W}^{q} \in \mathbb{R}^{2Dh \times 2Dh}$ as well while $\boldsymbol{b}^{q}, \boldsymbol{u} \in \mathbb{R}^{2Dh}$.

4.3.1 BiDAF output layer

This output layer is based on the original used in the BiDAF model proposed by Seo et al 2016. It applies the softmax function to the concatenation of the BiDAF output with the context embeddings to obtain start and end probability distributions:

$$\boldsymbol{s}_j^{start} = \boldsymbol{u}^{\mathrm{T}} [\boldsymbol{l}_i ; \boldsymbol{c}_i] \in \mathrm{R}$$
$$\boldsymbol{a}^{\mathrm{start}} = \mathrm{softmax}(\boldsymbol{s}^{\mathrm{start}}) \in \mathrm{R}^{N} \qquad [2]$$

The addition of the context embeddings allows the meaning of the vectors to increase.

$$\boldsymbol{s}_j^{end} = \boldsymbol{u}^{\mathrm{T}} [\mathrm{GRU}(\boldsymbol{l}_i) ; \boldsymbol{c}_i] \in \mathrm{R}$$
$$\boldsymbol{a}^{\mathrm{start}} = \mathrm{softmax}(\boldsymbol{s}^{\mathrm{start}}) \in \mathrm{R}^{N} \qquad [2]$$

4.4 Note on hyperparameters

The following hyperparameters were used in the highest-performing model. More complex models that integrated both fine-grained gating and a non-baseline output layer tended to exhaust memory; in those cases, batch size was reduced to 50 while learning rate was increased to 0.05.

Dropout	0.4
Learning rate	0.01
Batch size	100
Embedding size (both word and char)	200
Hidden size	200

One drawback of bi-directional attention flow was the tendency to overfit the answer to the question. After 4500 iterations, the model generated the following answer:

> QUESTION: **what theorem states that the probability that a number n is prime is inversely proportional to its logarithm ?**
> TRUE ANSWER: **the prime number theorem**
> PREDICTED ANSWER: **theorem**

This was presumably because the 'theorem' token had already appeared next to the question-word 'what' and generated a high relevance score through the $\check{c}$ vector, resulting in a high relevance score for the token that factored into both the start- and end- token distribution. To manage overfitting, I separately increased the dropout for the BiDAF output layer to 0.6 while other dropout values remained at 0.4.

5 Experiments

The experiment results showed that BiDAF, as expected, gave a significant improvement over the baseline. Fine-grained gating offered a slight improvement of about 2% over concatenating the word and character embeddings. Surprisingly, the answer pointer combined with bi-directional attention did not yield better results, with F1 scores plateauing around 0.4 and loss plateauing around 4 despite extensive debugging.

Table 1: F1 scores

Model configuration	Dev F1 score
Concatenation with baseline output layer	0.61
Fine-grained gating with baseline output layer	0.63
No character embedding with baseline output layer	0.59
Concatenation with BiDAF output layer	0.60
Fine-grained gating with BiDAF output layer	0.62
No character embedding with BiDAF output layer	0.57
Concatenation with answer pointer	0.46
Fine-grained gating with answer pointer	0.47
No character embedding with	0.46

answer pointer	

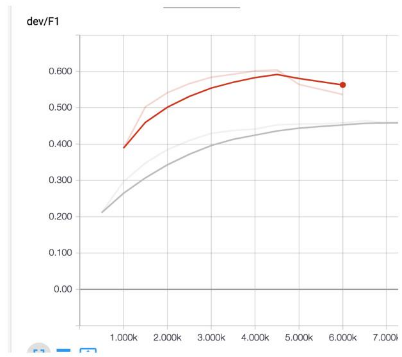

Fig. 1: Comparing baseline output (red) with answer pointer output (grey)

Analyzing model output example-by-example showed that the end tokens chosen corresponded quite well to the start tokens even using the baseline output layer. The answer spans chosen by models using the baseline output layer corresponded to word and sentence boundaries and generally corresponded well with grammatical phrases (e.g. selecting a whole noun phrase). They also attended well to other context tokens, as in the sample output below:

CONTEXT: […] however , if the forces are acting on an extended body , their respective lines of application must also be specified in order to account for their effects on the motion of the body . [Total length: 170 words]

QUESTION: when forces are acting on an extended body , what do you need to account for motion effects ?

TRUE ANSWER: respective lines of application

PREDICTED ANSWER: respective lines of application

Here, the phrase "forces are acting on an extended body" appears in the question as well as in the context, and the model is able to pick an exactly matching answer span presumably based on both the information from the C2Q representation and the Q2C representation — effectively allowing the model to take question-relevant portions of its context into account.

As for the answer pointer layer, it is possible that this approach generated worse results simply because it did not work with the BiDAF output, requiring different input such as the self-attention layer implemented in R-Net. I tried to implement the R-Net self-attention layer in addition to BiDAF, but this produced severe overfitting (0.8 F1 training score vs. a 0.4 F1 validation score) likely due to the increased model complexity. To avoid eliminating BiDAF altogether or scaling down model dimensions significantly, I did not explore this further.

The improvement produced by adding character CNNs was slight but to be expected; Seo et al reported a 0.03 boost in F1 scores from using concatenated character CNNs while Yang et al 2017 found a 0.017 boost in F1 scores from fine-grained gating over concatenation. [1] It is quite likely that SQuAD contains a low proportion of out-of-vocabulary tokens or tokens with unfamiliar morphology.

However, the concept behind fine-grained gates is of more general interest, since it can be applied beyond SQuAD (and may in fact be more useful in other contexts where out-of-vocabulary tokens are more frequent, such as comprehending highly technical texts with academic jargon) and also captures an interesting intuition about character- and word-level representations. Character CNNs are thought to enrich word representations for infrequent and out-of-vocabulary tokens, as well as supply morphological information. Intuitively, more frequent tokens should weigh character-level representations less than word-level representations, while less frequent tokens should rely more heavily on character-level representations. Fine-grained gating also allows twice the dimensionality of word- and character-level representations with comparable amounts of memory, since the representations are added rather than concatenated. The highest-performing model was able to use word and character embedding dimensions of 200 each before adding the weighted embeddings for a blended representation.

6 Further work

One possible avenue for further research is a simpler self-attention attention mechanism that allows each context token to attend to other context tokens — this would be analogous to RNet's simplified implementation of a bidirectional context-question attention mechanism, which allowed the model to reproduce the effects of the BiDAF layer without overcomplicating the model as a whole given the other complex attention mechanisms going on. In addition, an opportunity to test character-level representations for words and related mechanisms such as fine-grained gating on a more suitable dataset could allow progress on this particular word representation method. For instance, a dataset that uses a larger number of obscure or foreign words, such as an academic database or experimental literature.

7 Acknowledgements

Thanks to the whole teaching team for the amount of work that went into this engaging quarter, where the moments of pain and hair-pulling were almost always instructive. I learned a lot!

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

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

[3] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang and Ming Zhou Gated Self-Matching Networks for Reading Comprehension and Question Answering. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

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

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