Question Answering on the SQuAD Dataset

TG Sido
Department of Computer Science
Stanford University
osido@stanford.edu

Abstract

Question Answering (QA) is an increasingly important NLP problem with the proliferation of chatbots and virtual assistants. When a system has prior background knowledge, it can be queried with a question, which the system attempts to understand and answer using machine comprehension. This paper focuses on answering questions from the **S**tanford **Q**uestion **A**nswering **D**ataset (SQuAD) dataset which is comprised of paragraphs, questions, and their corresponding answers from Wikipedia. In order to achieve state-of-the-art results, past papers[2] have relied on a series of complex attention mechanisms, word vector embeddings, and novel deep learning optimizations. This paper focuses on the implementations of Bi-Directional Attention Flow (BiDAF), Dynamic Coattention Network (DCN), and smart span selection. Our ensemble of BiDAF and DCN achieves an F1 score of 58.7 and an EM score of 44.4 on the dev set.

1. Introduction

Question Answering (QA) systems have gained in popularity recently with chatbots and virtual assistants, including Alexa and Siri. Moreover, QA has democratized access to many facets of technology based on its ease of use. In this paper, we build a QA system for the **S**tanford **Q**uestion **A**nswering **D**ataset (SQuAD) dataset, which consists of thousands of adversarial questions that were crowdsourced from Wikipedia articles. Human Performance on SQuAD peaks at an EM score of 86.8 and an F1 score of 89.5 while PINGAN Gamma Lab's state-of-the-art system achieves an EM score of 83.4 and an F1 score of 85.9. Therefore, working on QA systems for SQuAD is worthwhile, as we still have progress to make. We take paragraphs and questions from those paragraphs as input and our system will output answers to those questions.

2. Related Work

Machine Comprehension (MC) involves understanding a context paragraph, so that questions from the paragraph can be answered. Pranav Rajpurkar and his team released a paper on SQuAD, which we refer to in order to better understand the structure of the dataset, the baseline of human performance, and its use of logistic regression, which achieved an EM of 51.0 and F1 of 51.0 [1]. Previously, early summarization - where the attention layer lacks a complete and fluent summary of the context in relation to the question plagued many older QA systems that relied solely only on Context-to-Question (C2Q) attention. In 2016, Minjoon Seo and his colleagues introduced **Bi-Directional Attention (BiDAF)** which combined attention flow from Context-to-Question (C2Q) as well as Question-to-Context (Q2C) with excellent results: EM of 67.7 and F1 of 77.3 [2]. In addition to this attention layer, they also use both a character and word embedding layer (we only use a word embedding layer) and a modeling layer. In our own paper, we implement BiDAF's attention layer as part of our strategy. Caiming Xiong's work on **Dynamic Coattention Networks (DCN)** also works to improve the attention layer by adding additional trainable states to both the context and the question, which over training time, help to correct from local maxima that correspond to incorrect layers [3]. DCN performs very well with EM of 65.4 and F1 of 75.6. Note that DCN also uses a dynamic pointing decoder to estimate the start and end indices of the span and a biLSTM as the initial encoder. Our model implements the DCN attention layer as part of our ensemble and uses a biGRU for our initial encodings. Danqi Chen's work on **smart span prediction**

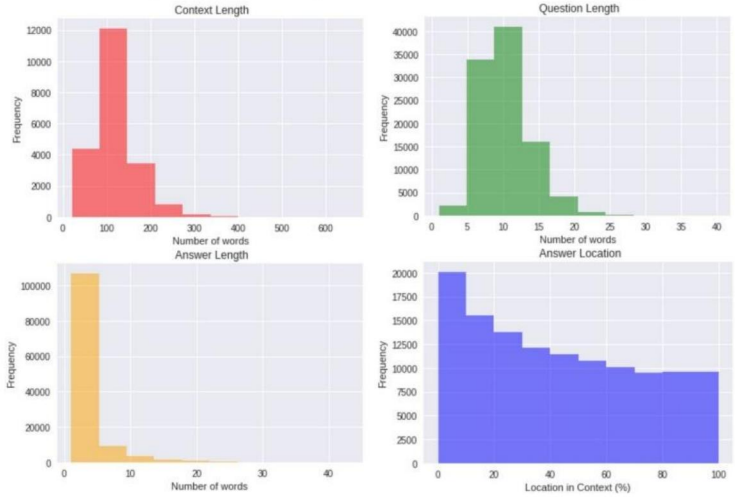
also inspired our own version of smart span prediction [4]. They select a span for the answer based where the start and end indices are at most 15 characters apart and where the joint probability of the deduced start and end indices being correct is maximized. They achieve an impressive EM of 69.5 and F1 of 78.8 with an array of other architectural features, including a 3-layer biLSTM for initial encoding. Our implementation uses a bitmask to enforce the distance between the start and end indices. Finally, Bhuwan Dhingra's work on using biGRUs as the intermediate steps of an attention layer provided useful insight into possible future work for our model [5].

3. Dataset

SQuAD comprises of comprises 107,785 examples from 23,215 paragraphs which come from 536 articles that follow this format:

P: The South African Schools Act of 1996 recognizes two categories of schools: "public" (state-controlled) and "independent"…
Q: What South African law recognized two types of schools?
A: South African Schools Act
Q: In what year was the South African Schools Act passed?
A: 1996
Q: Along with public schools, what type of school was recognized under the South African Schools Act?
A: independent
Legend: **P** (paragraph); **Q** (question); **A** (answer)

The train set has 87,599 examples and dev set has 10,570 examples, following a 90-10 split while test set is not revealed to the public. Data exploration reveals the following:

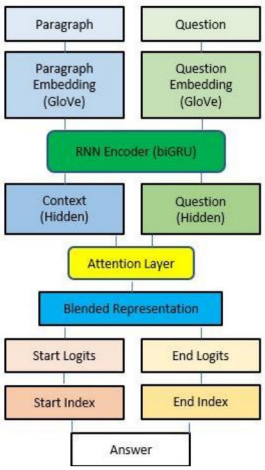


When examining the context length histogram, we see that most contexts are less than 300 words in length. Our initial model accommodated all context lengths up to 600 with padding, but this discovery led us to reduce our max context length to 300, which discards 112 paragraphs and associated questions but tremendously reduces computational cost, making it easier to train and evaluate our model. Next, when examining the location of where answers are in the original context, where 0% is the beginning of the context and 100% is the end of the context, we see that many answers lie in the beginning of the context with exponential drop-off behavior as we move to the end of the context. However, there is still substantial representation for every answer location within the context. Lastly, when viewing the answer length figure, we see that answers rarely are longer than 15 words. This informs our decision for smart span selection to enforce:

$$\text{startIndex} \leq \text{endIndex} \leq \text{startIndex} + 15$$

4. Methods

4.1 General Architecture



We build upon the baseline architecture shown above from CS224N[6] to build a more complex attention layer with BiDAF and DCN as well as smart span selection further downstream to determine the start and end indices.

4.2 Complex Attention Mechanisms

Given N different context hidden states (C) and M different question hidden states (Q), our baseline uses basic dot-product attention from context to question (C2Q). However, we can achieve better results when we allow attention to flow from question to context (Q2C) as well. Note for future reference that c_i refers to the ith context vector and q_j refers to the jth question vector. Our goal is to create a query-aware version of C, our matrix of context vectors.

4.2.1 Bi-Directional Attention Flow (BiDAF)[3]

BiDAF creates an NxM similarity matrix where S_{ij} has is a scalar score based on c_i and q_j. Note that this similarity matrix S, is in part created with a weight vector, w_{sim}, which is updated over time through backpropagation. To perform C2Q attention, we get α by taking a row-wise softmax of S to help form our a, which is a weighted sum of our question states, q_j .Then, for Q2C attention, we define m, where m_i is the max of the corresponding row in S, providing an attention distribution over context locations that highlights the most important parts of the context for the question. We then get β by taking the softmax of m to find the probability of each specific context location. This is then used to compute our Q2C output, c', which is a weighted sum of our context states and β. Finally, we get our final attention output, which combines both our C2Q and Q2C outputs by stacking c, α, element-wise multiplication of c and α, and element-wise multiplication of c and c'.

4.2.2 Dynamic Coattention Network (DCN)[4]

DCN, like BiDAF, uses both C2Q and Q2C. However, it differs from BiDAF in that it uses second-level attention to attend to the attention outputs as well. To achieve this, we first we use a trainable weight matrix, W_{mod} and vector, b_{mod} to derive modified question states: $q'_j = \tanh(W_{mod}\, q_j + b_{mod})$, which over training time, should select and weight different parts of the question states differently to aid in computing attention downstream. Then, we add trainable sentinel vectors to both our context and question matrices and compute an (N+1) x (M+1) affinity matrix L where $L_{ij} = c_i \cdot q'_j$. Note that the additional sentinel vectors are added to attend to none of the question and context states, which allows the sentinel vectors to build up a knowledge base over time to increase the performance of the attention computations downstream. To perform C2Q attention, we get α by taking a row-wise softmax of L to help form our a which is a weighted sum of our question states, q_j. Then, for Q2C attention, we get β by taking the column-wise softmax of L and use that to calculate our Q2C output, b, which is a weighted sum of our context states and β. Then we get our second-level attention, s, as a weighted sum of b and α. Finally, we stack s and a and feed it as input to a biLSTM -- as part of coattention encoding -- and return its output as our final attention output.

4.2.3 BiDAF + DCN Ensemble

In general, ensembling pools multiple approaches together for a single stronger approach. We do this here by stacking the attention outputs from both BiDAF and DCN as our final attention output.

4.3 Smart Span Selection

When choosing the start and end indices, our architecture uses the same logits distribution and selects the index with the highest probability independently. Independent selection can cause problems if the end index is before the start index, which is impossible. Additionally, since we discovered from data exploration that answers are typically no more than 15 words, we want to enforce: startIndex ≤ endIndex ≤ startIndex + 15. To achieve this, we use a bitmask of 1s where valid logits exist and 0s where they do not and perform element-wise multiplication of this bitmask and the logits distribution to enforce our rule. Since it uses argmax to find the index, zeroing out certain logits greatly lessens the probability of being chosen.

5. Experiments/Results/Discussion

5.1 Results

Model	Train F1	Train EM	Dev F1	Dev EM
Baseline (dropout = .15, context_len = 600)	61.1	49.2	39.4	29.1
Baseline (dropout = .15, context_len = 600, smart span selection)	62.3	51.2	39.5	28.6
BiDAF (dropout = .15, context_len = 600)	54.4	42.6	44.1	32.2
BiDAF (dropout = .20, context_len = 300)	66.3	54.6	45.2	33.1
DCN (dropout = .15, context_len = 600)	72.9	58.6	57.9	43.5
DCN (dropout = .20, context_len = 300)	72.5	58.8	58.1	43.3
BiDAF + DCN Ensemble (dropout = .20, context_len = 300)	73.3	60.1	58.3	43.2
PAML + BERT (Ensemble) (state-of-the-art)	**n/a**	**n/a**	**85.9**	**83.4**

5.2 Hyperparameter Choices

Data exploration and iteration informed our hyperparameter choices. With regards to *learning_rate*, we chose .001, which is slow but mitigates against divergence when finding the global minimum for the cost. For QA, this is ideal since this task there is a lot of granularity needed in the weight matrices to find the correct start and end indices. For *batch_size,* we initially chose 100 but switched to 32 due to memory and speed constraints on our dev machine. Generally, larger batches are better for lowering variance in cost estimates. Since most paragraphs (contexts) are less than 300 words in length, using 300 instead of 600 for the *context_len* hyperparameter provides tremendous speed improvements in training and testing. For *dropout*, we increased from .15 to .20 to increase regularization. As shown above in our results, regularization – due to increasing dropout probability or early stopping – generally provides a small but noticeable boost in performance. For *embedding_size*, which refers to the size of the GloVe word embeddings, we settle on 100 instead of 200 or 300. While word embedding vectors with larger sizes can encode more meaning, past papers have not seen huge performance increases from larger embeddings.

5.3 DCN vs BiDAF

DCN performs better than BiDAF due to the following: trainable sentinel states that are added to both the context and question that can utilize the growing knowledge base. Also, the additional biLSTM encodes more query-aware information into the attention layer by using for the first and second level attention layers.

5.4 Smart Span Selection

In order to test smart span selection in isolation, we ran it with the baseline architecture to compare to our initial baseline. We see a minimal increase in dev F1 but a decrease in dev EM. More investigation will need to be done here to improve this feature.

5.5 Error Analysis on our Ensemble

Here, we will cover a few key examples where our ensemble model failed:
Note that **green text** is true answer, magenta background is predicted start, red background is predicted end, underscores are unknown tokens.

Example A:

the American broadcasting company (abc) (stylized in its logo as abc since 1957) is an american commercial broadcast television network that is owned by the _disney–abc_ television group , a subsidiary of disney media networks division of the walt disney company . the network is part of the big three television networks . the network is headquartered on **columbus avenue and west 66th street** in manhattan , with additional major offices and production facilities in new york city , los angeles and burbank , california .
QUESTION: on what streets is the abc headquarters located
TRUE ANSWER: columbus avenue and west 66th street; PREDICTED ANSWER: columbus avenue
F1 SCORE ANSWER: 0.500; EM SCORE: False

Analysis: Our system does find the correct start index and finds an incorrect end index that is still within bounds of the true answer. This is likely an early summarization issue, as mentioned previously, we need to investigate our ensemble's attention layers and possibly investigate a dynamic pointer feature to mitigate against this.

Example B:

most _platyctenida_ have oval bodies that are flattened in the _oral-aboral_ direction , with a pair of _tentilla-bearing_ tentacles on the aboral surface . they **cling to and creep on surfaces** by _everting_ the pharynx and using it as a muscular " foot " . all but one of the known _platyctenid_ species lack _comb-rows_ . _platyctenids_ are usually cryptically colored , live on rocks , algae , or the body surfaces of other invertebrates , and are often revealed by their long tentacles with many _sidebranches_ , seen streaming off the back of the _ctenophore_ into the current .
QUESTION: what do platyctenida use their pharynx for ?
TRUE ANSWER: cling to and creep on surfaces; PREDICTED ANSWER: a muscular " foot
F1 SCORE ANSWER: 0.000; EM SCORE: False

Analysis: Our system fails completely with this paragraph and question, and it is likely due to the prevalence of unknown tokens (shown with underscores). Since Wikipedia has obscure words (scientific terms in this case) that are not in the GloVe vocabulary, our model has a hard time creating a useful attention layer. Using a character-level CNN to augment the initial word embeddings would help here.

6. Conclusion/Future Work

As mentioned in the introduction, human performance on SQuAD peaks at an EM score of 86.8 and an F1 score of 89.5 and state-of-the-art systems still have much ground to cover to reach that level of performance. From our investigation of the BiDAF and DCN systems, we see that a strategic attention strategy is vital for all high-performing models. Additionally, we see that there is a lot of room for creativity in approaching this task. Future work could include implementing smart span selection with an LSTM to condition probability of end index on the start index; using a character-level CNN to augment the GloVe word embeddings, mitigating against out-of-vocabulary words; experimenting with averaging, adding, or max-pooling the forward and hidden states from the RNN Encoder.

7. Contributions

TG Sido worked solo on this project. He implemented BiDAF and DCN, and smart span selection by following the equations set forth in their respective papers and implementing them in TensorFlow in this repository. He also wrote the visualization code in a jupyter notebook.

References

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

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

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016.

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

[5] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. arXiv preprint arXiv:1606.01549, 2016.

[6] CS224N. https://web.stanford.edu/class/cs224n/default_project/default_project_v2.pdf

Acknowledgements

We would like to thank Pedro Garzon and Steven Chen for helpful advice and insights regarding this project.