Attention and CNN empowered BiDAF for SQuAD

Haojun Li
Department of Computer Science
Stanford University
haojun@stanford.edu

Abstract

The SQuAD dataset introduced in Rajpurkar et al. [2018] formulate the Question and Answer problem as a span prediction problem with adversarially written unanswerable questions. This formulation enabled a vast amount of research in Question and Answer systems, and many recent research efforts are focusing on achieving and surpassing human performance. In this project I aim to establish a better baseline than the one established for us, and improve upon it by incorporating Convolutional Neural Networks with Self-attention mechanisms as in Yu et al. [2018] into the RNN-based BiDAF model, as well as experimenting with a novel (yet under-performing) multi-headed BiDAF attention model. I have shown that by combining CNN-based and RNN-based encoders, the model performs better than those encoders alone, and significantly better than the baseline model. On the other hand multi-headed BiDAF attention models perform significantly worse due to overfitting and other fundamental model restrictions.

1 Introduction

Question and Answering (QA) Systems is not only a widely researched subject in NLP but also used extensively in the real world. Many early systems frame it as an information retrieval problem, where they look for word similarities and return the most relevant passages (much like a search engine) while developing sophisticated similarity metrics such as Term Frequency Inverse Document Frequency (TF-IDF) as in Ramos et al. [2003]. With the increasing ability of storing and processing massive amount of data, research has shifted into building massive data storage and using NLP techniques to extract keywords and indices. However, these systems are only able to retrieve a document most relevant to a question, rather than the answer itself. This means questioners can only retrieve the most relevant document and read through it themselves, rather than receiving a brief and succinct answer to the question. The need for simple answers gave rise to Knowledge Bases (KBs) such as Freebase as in Bollacker et al. [2008], which stores information in key-value tuples in a categorical and structured way. This shift prompted researchers to develop more and more sophisticated systems to do Knowledge Base extraction, building semantic parsers and entity recognition systems to extract and distill information that can be stored in KBs. However, this system have fundamental restrictions, such as the fact that real world data is not structured, and answers to a question might not be easily categorized as a key-value pair, especially questions that require logic.

Recent developments in neural techniques gave rise to a host of advanced systems that yield dramatic improvements with only minimum engineering effort. This revolutionized the field of NLP and allowed for more complex QA to be performed. The SQuAD dataset introduced as part of Rajpurkar et al. [2016] attempts to structure this problem. More specifically, each entry of the SQuAD dataset contains a context paragraph, a question, and an answer, where the answer can be found in the context as an exact match. This reduces to a span prediction problem, where given a context and a question, QA systems should be able to predict the start and end positions of the answer in the context paragraph. Many techniques works incredibly well on the SQuAD dataset, from using Bi-Directional LSTM to Convolutional Neural Networks with attention, some even surpasses human performance.

The dataset is later augmented with unanswerable questions written adversarially as in Rajpurkar et al. [2018] to make sure these systems are not susceptible to attacks. This project aims to combine some of the techniques used by previous researchers and improve upon a pre-established baseline. Specifically, I attempt to first establish a better baseline with character level embeddings, and then creatively incorporate convolutional neural networks as introduced in Yu et al. [2018] to improve upon it. This has yielded great results dramatically improving the baseline model provided to us. I also attempted to use a novel architecture of multi-headed BiDAF model to improve upon this newly established baseline. This has yield very limited results due to overfitting and fundamental model restrictions.

2 Related Work

Many researchers have developed complex architectures to solve the Question and Answer problem structured by the SQuAD dataset. The first system introduced by Rajpurkar et al. [2016] frames this problem as a multi-class classification problem, thus the paper uses dependency and tree parsers with logistic regression model to perform question and answer matching. This has extremely limited results, but did significantly improve a sliding window baseline. The system that did significantly well, and thus is the pre-established baseline, is the BiDAF model introduced in Tuason et al.. This model uses Bi-directional LSTM with a bi-directional attention flow network to predict start and end positions of the answer in context. Later models have improved on this architecture by incorporating self-attention as introduced in Wang et al. [2017] after the BiDAF attention layer. More radical improvements have been made upon the BiDAF model using depth-wise separable convolutional neural networks with self-attention and positional embeddings as introduced in Yu et al. [2018], and this is the first model that achieved better scores than human performance. This network does away with recurrent networks and replace them with highly parallelizable convolutional networks, not only improving the training speed but also the results, which researchers attribute to more iterations of hyper-parameter tuning. However, this network does not perform quite well in the SQuAD 2.0 dataset with unanswerable questions.

With the introduction of advanced pre-trained contextual embeddings such as ElMo introduced in Peters et al. [2018] and BERT introduced in Devlin et al. [2018], these dramatically improved the performance of almost all architectures currently in use by replacing non-contextual word embedding such as GloVe introduced in Pennington et al. [2014] with these contextual embeddings. However, this project will not be using these contextual embeddings as most of the effort would be spent integrating them into existing code.

3 Approaches

I have attempted many approaches and only 2 are worth mentioning here. The first model is combining the encoder block as introduced in Yu et al. [2018] with current Bi-directional encoder in various ways. The second model is a new model which uses multiple BiDAF attention similarity matrices. Details about the models are explained below.

3.1 Baseline

The baseline of this project is an improvement on the model provided to us by the instructors. I have added character embeddings and fine-tuned the BiDAF model to achieve reasonable results, which will be used as the new baseline of this report. Details of this model is in the project handout so they will not be included here.

3.2 Attended CNN + RNN Encoding

The first and the most fruitful model I have implemented borrows the Encoder Block from the QANet paper [Yu et al. [2018]]. The encoder block is shown in the picture 1, taken directly from Yu et al. [2018].

My implementation of the encoder block differs from the QANet as referenced by figure 1. Details about the original implementation can be accessed in Yu et al. [2018]. In the interest of space only the difference in adaptations are listed below.

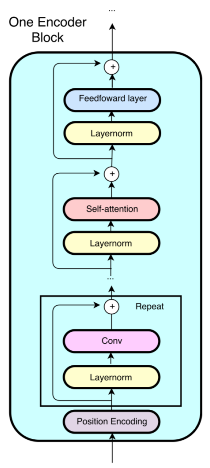

Figure 1: The encoder block taken from Yu et al. [2018]

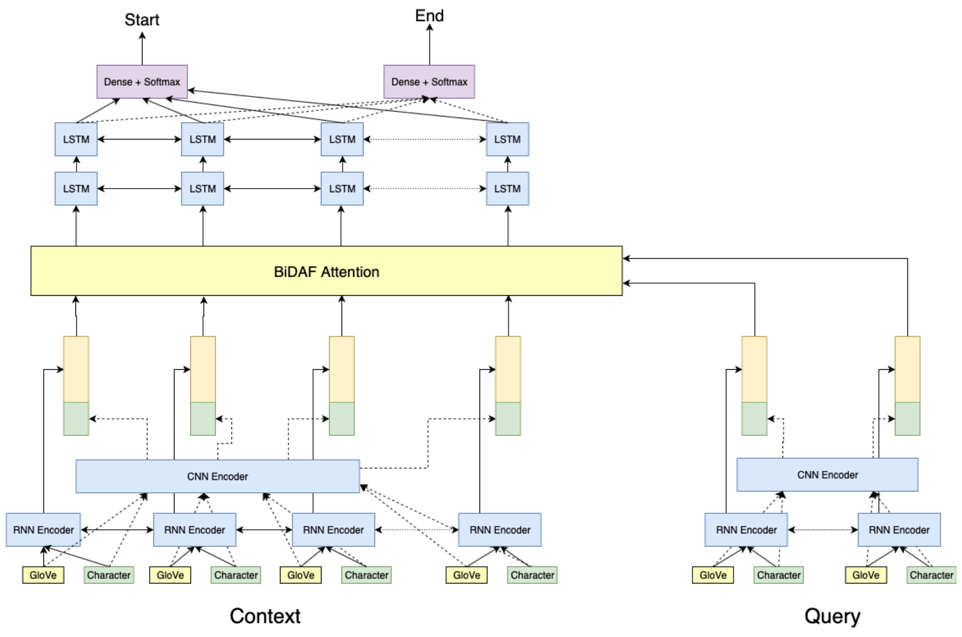

Figure 2: CNN+RNN Encoder BiDAF

1. My encoder block does not have positional encoding. In my experiments positional encoding did not add much improvements.
2. The number and size of Convolution Layers is generally smaller than proposed by the QANet paper due to limited computational capacity. The input and output of the convolution layer have the exact same shape. The output size is padded to have the same shape as the input window size using formula $p = \frac{f-1}{2}$ where p is the padding needed on each side of the input and f is the kernel size of the CNN. This convolution layer is repeated multiple times with residual connections between each one. This idea is proposed by He et al. [2015] to allow for deeper neural networks without the problem of vanishing gradients. Note that my CNN layers are not depth-wise separable since I reduced the size and number of CNNs too much (due to memory limitations) for this to be effective.
3. The self attention layer is implemented using the same idea as Vaswani et al. [2017]. This allows the convolution layers to "access" other locations of convolution output by calculating, at each position, a weighted sum of the all other position output of this convolution layer. This is done by feeding the input vector into 3 linear layers Q, K, and V. Then, it calculate the attention as $A = softmax(Q, K^T)V$. This layer also has a residual connection, which means that the final output of the layer is $O = x + softmax(Q, K^T)V$. I also implemented the multi-headed self attention to improve performance. I implemented all of these since I did not realize that I can use existing code from GitHub.

This encoder block is *not* used as defined in the QANet paper. Instead, I experimented with using the encoder block in various creative ways. Most importantly, I used this encoder block (shown as CNN Encoder in figure 2) to encode the input word and character embeddings into a fix sized feature vector for each position of the question and context respectively. Then, I concatenated this feature vector with the original LSTM encoding from the BiDAF paper, forming one single feature vector for each position of the question and context respectively. The final architecture is shown in figure 2. I hypothesize that the RNN encoder is good at encoding temporal information, while stacked CNNs are better at encoding context-invariant local information. Together, they would form a better encoding for the Bi-DAF attention layer to perform similarity scoring. This is something I *came up on my own* after a lot of failed experiments, though I'm sure I'm not the first one. Note that we share weights between the context and query CNN encoder, meaning that we use the same encoder for both context and query.

I also swapped out the 2 bi-directional model LSTM with the encoder blocks but they did not perform as well as I expected. In the interest of space these failed experiments are not discussed.

3.3 Multi-head BiDAF

Another architecture that I experimented with is the multi-headed BiDAF attention layer. The baseline code uses BiDAF layers to calculate attention scores and transforms the context and query into a single feature vector that "encodes" the context with respect to the query by calculating a similarity matrix. I hypothesize that having a single similarity matrix is insufficient in calculating the attention scores, specifically because I hypothesize that there could be multiple "similarities" between the context and the query that a single matrix is unable to capture. Thus, I used the following formula to calculate my similarity matrix (assuming we have hidden states c_i and q_i forwarded through k linear layers to get $c_i^{(k)}$ and $q_j^{(k)}$):

$$S_{ij}^{(k)} = w_{\text{sim}}^{(k)^T}[c_i^{(k)}; q_j^{(k)}; c_i^{(k)} \circ q_j^{(k)}]$$

Then, I calculate multiple attention outputs

$$\bar{S}_{i,:}^{(k)} = \text{softmax}(S_{i,:}^{(k)}) \qquad a_i^{(k)} = \sum_{j=1}^{M} \bar{S}_{i,j}^{(k)} q_j^{(k)}$$

$$\bar{\bar{S}}_{:,j}^{(k)} = \text{softmax}(\bar{S}_{:,j}^{(k)}) \qquad b_i^{(k)} = \sum_{j=1}^{N} S_{i,j}^{'(k)} c_j^{(k)}$$

$$S^{'(k)} = \bar{S}^{(k)} \bar{\bar{S}}^{(k)^T}$$

Then, I attempted 2 different ways of combining these attention outputs

1. I took the max of these attention outputs across all attention heads for each attention output. Specifically $c_{ij} = \max_k(c_{ij}^{(k)}),\ q_{ij} = \max_k(q_{ij}^{(k)}),\ a_{ij} = \max_k(a_{ij}^{(k)}),\ b_{ij} = \max_k(b_{ij}^{(k)})$. This means that each feature of each output is the max of that feature across all attention heads.
2. I used a linear layer to map the concatenated features of all attention heads back to a feature vector of the same size as the input feature size, same as Vaswani et al. [2017]. This is done using the formula $a_i = W_a[a_i^{(1)}; a_i^{(2)}, ..., a_i^{(k)}] + bias_a$ and similarly for b_i, c_i, q_i.

This has also yielded limited results. Details about the results are discussed later.

4 Experiments

I have run numerous experiments both on Azure as well as on my personal GPU. Most of the experiments failed to improve the baseline and only some have succeeded. Only the notable experiments will be discussed here.

4.1 Data

The data and pre-processing steps are unchanged. The input word and characters are transformed into indices, with word indices corresponds to the dictionary index of the word's GloVe embeddings. For more details about how data is pre-processed, please refer to the project handout.

4.2 Evaluation Method

The evaluation method used is the same as all SQuAD papers, by using EM (Exact Match) and F1 scores. The evaluation code is also unchanged.

4.3 Experimental Details

All of the following experiments are run on an Adam Optimizer with learning rate 0.001 and default β parameters ($\beta_1 = 0.9$ and β_2=0.999). This change is motivated by Adam's faster convergence speed as demonstrated by Kingma and Ba [2014].

1. Bi-DAF Fine-tune: First I implemented character level embedding with CNNs of kernel size 5 and character embedding size 50. The output of the padded CNN layer is of size 256 with max pooling across all windows, which is then forwarded through a highway network and concatenated with word-level embeddings forwarded from a different highway network, forming a total word embedding of size 512. The model hidden size is set to 128 with a mini-batch size of 64, which means hidden size for cell and hidden layers for LSTM of the entire biDAF model is set to 128. We have a drop out rate of 0.3.
2. Attended CNN+RNN: With the same word and character level embedding as the previous experiment, I implemented the Encoder Block as specified in previous sections with one 128 filter CNN and 128 feature size for the feedforward layer. I have a 7-head self attention layer within the Encoder Block with hidden size 50. I have a mini-batch size of 80 with drop out rate of 0.3. Then, I increase the number of heads to 8 and stacked 4 CNNs on top of each other within the Encoder Block, with a mini-batch size of 75. Note that the output of the embedding layer is a 512 feature vector for each word, which is immediately mapped to 128 feature vector for each word using a convolution layer before being fed into the Encoder Block. All other layers are the same as the previous model with changes to the input feature size of Bi-DAF attention layer (from 256 to 384) due to the concatenation of LSTM and Encoder Block outputs.
3. Multi-BiDAF: With the same word and character level embedding as previous experiments, I keep the embedding encoder and model encoder the same as baseline BiDAF, but added 3 heads to the BiDAF attention layer as defined in the previous sections. I have the same 128 hidden sizes throughout the model.

4.4 Results

The results are summarized in the table 1. Note that all experiment results are development split results unless otherwise specified. I'm doing the **Non-PCE** leaderboard, and have uploaded my results as **Simple Transformers**. As shown in the table 1, most of the experiments improved upon

Table 1: Results from experiments

Model	Modification	EM	F1
BiDAF Baseline		55	58
BiDAF character Embedding	fine-tuned new baseline	61.18	64.2
Multi-BiDAF	feed-forward shrink	58.23	61.14
Multi-BiDAF	max pool	60.46	63.59
Encoder Block only		55.96	59.71
Encoder Block + RNN Encoder		61.92	65.02
Encoder Block + RNN Encoder	Stacked + more heads	64.779	67.87
Encoder Block + RNN Encoder	Test split results	**62.671**	**66.045**

the original baseline, but only Encoder Block + RNN Encoder was able to improve upon the newly established baseline with character embeddings and fine-tuning. The "Encoder Block only" model performed much worse than expected. I suspect that is because convolutional layers are unable to retain temporal information like RNN encoders, and thus feeding only context-invariant information to the BiDAF attention layer restricts the model's ability to utilize contextual evidences.

I suspect that the Multi-headed BiDAF not only lacks in representational power, but the max pooling mechanism actually disadvantages the model by discarding relevant weak signals. On the other hand, using a linear layer to map the multiple features from multiple attention heads overfit the data very easily starting from 1.5 million iterations, with dev NLL steadily increasing past 1.8 million iterations.

5 Analysis

Here we will go in depth in analyzing what the model did well and what the model does not do well. Note that I will be comparing my models against the new baseline of fine-tuned BiDAF with character embeddings as this makes more sense than comparing it against the naive baseline given to us in the starter code. All analysis is done on the Dev split of the data.

5.1 CNN + RNN Encoding

The CNN Encoding Block + RNN is the best performing model out of all the experiments run. However, the main weakness still lies in the fact that it is unable to detect unanswerable questions. The confusion matrix of Answer vs No Answer is shown in figure 3. As we can see from the confusion matrix plot, the system is very able to recognize that there is an answer to a question when there is truly an answer. However, the system is less able to predict "No Answer" when there is truly no answer. On the other hand, the model performed much better than the fine-tuned BiDAF with character embedding in both recognizing when there is no answer (shown in 4) and answering the question when there is an answer (higher EM scores). I suspect that this is because the stacked convolutional neural network with self-attention is able to encode more relevant information and provide BiDAF attention layer with better understanding of the context and question.

Another issue that I suspected is that even with fixed window CNN embeddings, the mdoel is unable to predict correct answers in long contexts. This is illustrated in figure 5. As we can see, the model is significantly better in making predictions for shorter sentences, with about a 70% success rate, while performing worse in longer context paragraphs, with about a 40% to 50% success rate.

Indeed, when picking out some examples of wrong predictions, we can see that the model is unable to recognize long term dependencies as shown by this example below:

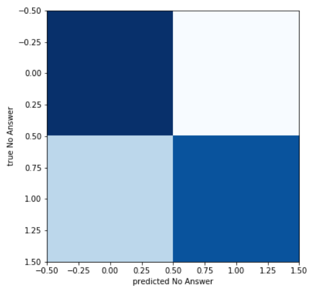

Figure 3: CNN + RNN Encoding

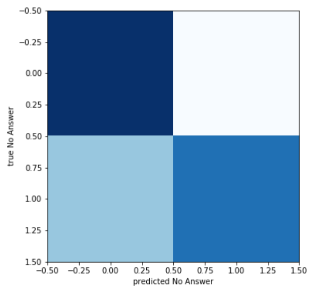

Figure 4: Fine-tuned BiDAF with character embeddings

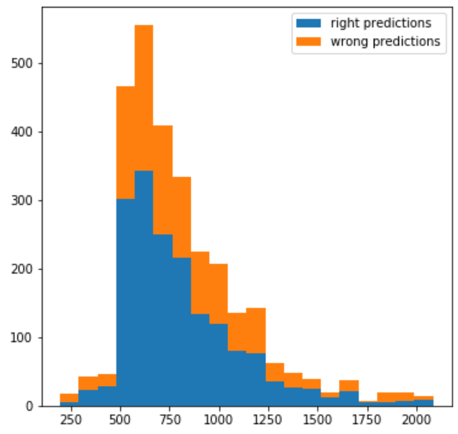

Figure 5: Histogram of length of context paragraph

- Question: What percentage of electrical power in the United States is made by generators?
- Context: The final major evolution of the steam engine design was the use of steam turbines starting in the late part of the 19th century. Steam turbines are generally more efficient than reciprocating piston type steam engines (for outputs above several hundred horsepower), have fewer moving parts, and provide rotary power directly instead of through a connecting rod system or similar means. Steam turbines virtually replaced reciprocating engines in electricity generating stations early in the 20th century, where their efficiency, higher speed appropriate to generator service, and smooth rotation were advantages. Today most electric power is provided by steam turbines. In the United States 90% of the electric power is produced in this way using a variety of heat sources. Steam turbines were extensively applied for propulsion of large ships throughout most of the 20th century.
- Answer: N/A
- Prediction: 90%

The "90%" refers to electric power generated "this way", which refers to "steam turbines". However, the model is unable to make this kind of connection, and naively matched "electric power in the United States" with "In the United States 90% of the electric power ...". This is a recurring theme of this kind of models, and more work needs to be done to improve this model and allow it to form long term dependencies.

5.2 Multi-head BiDAF

The multi-headed BiDAF turns out to be an absolute failure. The multi-headed BiDAF with a linear layer mapping all the features of all the attention heads to fixed sized output features introduced too many parameters and started to over-fit the data very early. Taking the max of each feature across attention heads turns out to perform a little worse than newly established baseline as well. However, the confusion matrix of this model and the new baseline model is very similar, with baseline model perform a little better when recognizing no answers. I suspect this is because taking the max of each feature across attention heads specifically gets rid of weak and negative signals from the bottom layers, preventing the upper layer network from receiving signals that is crucial in making a decision. One such example is shown below:

- Question: Unsurprisingly, the mujahideen's victory with the Soviets in the 1980s succeeded to produce what?
- Context: In Afghanistan, the mujahideen's victory against the Soviet Union in the 1980s did not lead to justice and prosperity, due to a vicious and destructive civil war between political and tribal warlords, making Afghanistan one of the poorest countries on earth. In 1992, the Democratic Republic of Afghanistan ruled by communist forces collapsed, and democratic Islamist elements of mujahdeen founded the Islamic State of Afghanistan. In 1996, a more conservative and anti-democratic Islamist movement known as the Taliban rose to power, defeated most of the warlords and took over roughly 80% of Afghanistan.
- Answer: N/A
- Prediction: Afghanistan one of the poorest countries on earth

Interestingly the model is able to make long term dependencies, such as recognizing that "due to vicious..." is within 2 commas which means that "making Afghanistan..." is attached to "the mujahideen's victory against...". However, "with" and "against" might have had positive and negative signals. Since it is taking the max of all the input signals, it might have discarded important information such as this one by only taking the strongest positive signal, thus crippling the modeling layer in the amount of information it can use.

6 Conclusion

I have implemented and experimented with 2 different architectures (and many others) with varying degrees of success. First I realized that the baseline model provided to us can be easily improved by adding character embeddings and fine tuning its parameters. This shows that in real life one can easily improve a model with simple parameter-tuning and incremental improvements. Comparing to a tuned BiDAF model, incorporating an Encoder Block to the Embedding Encoding layer inspired by Yu et al. [2018] is able to greatly improve the new baseline. Not only does it recognize much more No Answer questions, it is able to do better in Exact Match scores on answerable questions as well. Multi-headed BiDAF which is inspired from multi-headed attention as in Vaswani et al. [2017] performed significantly worse than baseline and my expectation. The reason seems to be a combination of over-fitting and fundamental model restrictions.

7 Future Works

Still, there are a lot of work that could be done. One thing is that I did not use l2 regularization on the weights of the model, simply because I did not have time to experiment with these weight decays. In the future, I would like to experiment with l2 regularization to reduce over-fitting. Other work include adding self-attention on the outputs of BiDAF attention layer and the context encoding output as in Wang et al. [2017]. To directly improve the model, I would consider Transformer-XL as in Dai et al. [2019], which has shown improvements in helping the model learn longer term dependencies, which my model suffers from the most.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860, 2019. URL `http://arxiv.org/abs/1901.02860`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018. URL `http://arxiv.org/abs/1802.05365`.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL `http://arxiv.org/abs/1806.03822`.

Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142, 2003.

Ramon Tuason, Daniel Grazian, and Genki Kondo. Bidaf model for question answering. *Table III EVALUATION ON MRC MODELS (TEST SET). Search Zhidao All.*

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198, 2017.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL `http://arxiv.org/abs/1804.09541`.