Question Answering with Gated Attention and Multitask Learning - Option 3 (Graded)

Stanford CS224N {Default} Project

Lam Wing Chan
Adobe
lamwingc@stanford.edu

Mingyang Ling
Google
mingyanl@stanford.edu

Abstract

Recently, there is a growing interest in the research of machine text comprehension and automatic question answering. Many state-of-the-art QA models consist of two components: recurrent neural networks to encode sequential inputs and structures, and attention mechanisms to deal with long term interactions of context and query words. In this paper, we started with the BiDAF, and included character-level embedding, self attention mechanism with gated residual blocks to enhance performance. Then, we discussed QANet's non-recurrent architecture, and applied character-level embedding and residual attention moduel to QANet. We also unified the output layer with single branch to equally learn the start and end positions. To further improve the performance, we propose multi-task learning with ensemble inference. Along with optimization and hyperparameters tuning, we have obtained competitive results on the SQuAD 2.0 dataset.

1 Introduction

In recent years, there is a growing interest in the research of machine text comprehension and automatic question systems. Many state-of-the-art QA models consist of two fundamental components: recurrent neural networks to encode sequential inputs and their structures, and different attention mechanisms to deal with long term interactions of context and query words.

Despite the success of many similar models on various benchmark datasets, slow training and inference as a result of the sequential nature of recurrent neural networks remains a fundamental challenge these models have in common. With long texts, expensive computation resources are required, leading to high turnaround time for carrying out iterative experimental trials. In addition, such high costs of training prevents researchers from training, evaluating, and scaling their model on large datasets. Slow inference also makes deploying novel models in production impossible, preventing these models from being exposed to real life applications and adapting to daily use cases.

In this paper, we attempt to discover new model architectures that replace the recurrent nature of many state-of-the-art models with feed-forward networks without harming performance. We aim to discover a novel non recurrent model architecture that can complete training and evaluation in a shorter time. Our explorations center around two novel architectures - the Bi-Directional Attention Flow network (BiDAF) [1] and the QANet [2]. Building upon these two architectures, we will incorporate character-level embeddings, self attention mechanism, gated residual block into these models. We will also reformulate QANet's loss function to achieve multiasking, and unify output layer's branches to reinforce better generalization. Coupled with fine-tuning of hyperameters, we will conduct a thorough analysis of models' performances and present out findings. We will discuss how these proposed approaches speed up learning process and analyze qualitatively how various components contribute to improved performances.

2 Related Work

Previous works have explored incorporating attention mechanisms [1] in a multi-stage hierarchical architecture to represent context and query inputs at various levels of granularity. Yu at al. proposed a bi-directional attention flow mechanism to obtain query aware context representation. The attention mechanism computes attention values at every time step, coupled with outputs from previous layers, information loss due to early summarization can be minimized. The attention mechanism is also memory-less, meaning that the attention value is computed based on only the query and context paragraph at the current time step, and does not depend on the attention value from a previous time step. This allows the attention layer to learn the attention between the input query and the context paragraph, while the modeling layer focused on learning the interactions between the query-aware context paragraph representation. It also prevents attention at current time step to be affected by an incorrect attention computation from previous time steps.

The contribution of [2] to the question answering research can be summarized as: Altering existing model architectures to replace recurrent neural networks with convolutions and self attention. This speed up training and interference procedures and makes such model extremely valuable for preparing QA models to scale up to large datasets and deployment to production. Yu et al. [2] obtained context and question embeddings by concatenating pre-trained GloVe and character embeddings. Instead of recurrent networks as building blocks of the context or question encoder, Yu et al. [2] carried out depth wise convolution and self attention using the multi-head-attention mechanism [3]. Similarly, the modelling layers in [1] are replaced by the new adaptations of convolutions instead of recurrent networks. Such implementation proves to be the key to improved memory and time efficiency. The quick turnaround time enables easily carrying out future experimentations and parameter tuning, and one important advantage that came with the improved architecture is the flexibility of training model with extra data.

Zhang et al. [4] argues that machine reading comprehension systems should be able to distinguish that there exists no answer to a question given a passage and refuse to proceed with answering. They proposed the addition of a verification module to verify that focuses on early detection of unanswerable questions. The retrospective reader integrates a two step strategy that involves reading and verification. It first conducts a scan through the passage and question to draw brief insights into the interactions of passage and question, forming an initial judgement on whether the question is answerable. Then an intensive comprehension of the judgement is carried out to to produce candidate answer spans, verify whether the question is answerable and provide the final answer.

Hu et al. proposed the Reinforced Mnemonic Reader [5] for carrying out reading comprehension tasks, which serves as an enhancement to previous attention mechanisms. They proposed a re-attention mechanism to refine current attentions, through accessing past attentions that has been memorized temporally in a multi-round alignment architecture. This aims to avoid attention redundancy and attention deficiency. In addition, they proposed a novel optimization approach to carry out dynamical critical reinforcement learning to encourage predicting a more acceptable answer to address the convergence suppression problem.

3 Approach

Baseline As a baseline for our task, we conducted experiments using the default project sample code, which implements the Bi-Directional Attention Flow network BiDAF [1] without using character embeddings. This model adopts a hierarchical multi-stage architecture that models the contextual paragraph with various levels of granularity. This baseline model contains the word embedding layer, contextual embeddding layer, attention flow layer, modeling layer and the output layer. The word embedding layer maps each word to a high dimensional vector space by looking up the corresponding GloVe word vectors [6]. The contextual embedding layer uses bi-directional long short term memory network (LSTM) [7] to model temporal interactions between words in the context/query paragraph respectively. It introduces bi-directional attention flow: from the input question to the context paragraph, and from the context paragraph to the question, to obtain the query-aware context representation. The modeling layer encodes the query aware representation of the context paragraph, and captures the interactions among these words given the query.

Character-Level Embedding We further added character-level embeddings to encapsulate information of input query and context paragraph as described in BiDAF [1]. The character embedding layer maps each word in the context and the query to a high-dimensional vector space, utilizing the algorithm discussed in [8]. All the character embeddings are concatenated, yielding a $m \times e_{char}$ tensor. This is fed into a 1D Convnet (CNN) with Relu and Maxpooling over the entire width, resulting in a fixed-size vector for each word. Thus, each word in the context and query are transformed into both word-level and character-level embedding. Then two embeddings are concatenated as input to generate a contextual embedding through Highway network, as illustrated in Figure. 1.

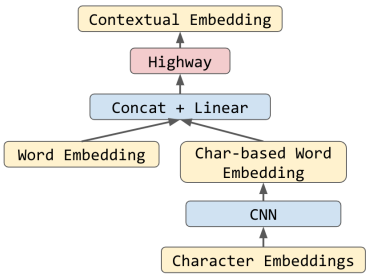

Figure 1: Character Level Embedding

The addition of this layer is significant for the understanding of the context paragraph and the query, as it extracts important features at different levels of granularity - analogous to the multi-stage feature computation through convolution neural networks in the field of computer vision tasks. The introduction of character embeddings solves a fundamental problem: out-of-vocabulary words are not taken into account when training the model. This feature extraction pipeline enables model to understand not only the semantic of words, but also the structural formation (spelling) of these words.

Self Attention Knowledge of the entire context outside of a surrounding window is essential to answer inference. However, recurrent neural models can only memorize information within a limited context window, and this leads model being unaware of important clues in other parts of the context paragraph. RNNs factor out computation into sequence of steps that aligns to each position in the input sequence, generating sequence of hidden states that depends on previous hidden states and the input at current position. This dependency on previous hidden state prevents parallelization and leads to memory constraints for long sequence lengths. We eliminated such shortcoming of existing baseline method by adopting a Self Attention mechanism proposed in Attention Is All You Need [3]. It describes an attention function that maps a query and a set of key-value pairs to an output. The output is a weighted sum of values, where the weights associated with each value is a compatibility measure of the query and the corresponding key. In our implementation, we have adapted the scaled dot-product attention on top of the Bidirectional Attention unit. The attention formulates as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

This method refines the representation by matching the paragraph against itself. Using the question aware context paragraph representations, we matched this representation against itself to obtain attention information. This attention mechanism allows modelling global dependencies among words in the inputs and output sequences regardless of their distances. Then we used attention to aggregate information from all word in the context paragraph. As a result, the Self Attention mechanism can effectively encode information from the whole paragraph.

Gated Residual Attention To improve on top of the self attention base architecture, we have incorporated a Gated Residual block to fuse the input representation with the self attention output as proposed in [5]. The gate controls how the attention output interact with the model, and how much information from the original input gets propagated through. This mechanism is analogous to the Highway network architecture in the character embedding layer, where only a fraction of the input gets propagated through, adjusting the influence and contribution from attention output. Given the original input x and the output a from the above Self Attention module, the final output can be

computed as follows:

$$\begin{aligned} \tilde{x} &= \text{Tanh}(W_r[x;a;x \circ a;x - a]) \\ g &= \text{Sigmoid}(W_g[x;a;x \circ a;x - a]) \\ o &= g \circ \tilde{x} + (1 - g) \circ x \end{aligned} \tag{2}$$

The above serves as a heuristic fusion function that fuses attentive information into the input, where $\circ$ denotes element wise multiplication. As a result, the output o from fusion is a linear interpolation of the input x and the intermediate output $\tilde{x}$, while the gate g controls the composition degree. The complete self attention module is illustrated in Figure. 2.

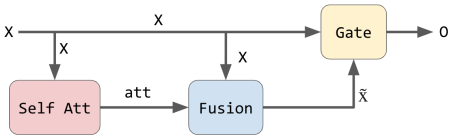

Figure 2: Gated Residual Attention

QANet In addition, we have implemented the QANet architecture [2]. QANet adapts ideas from the Transformer [3] and applies them to question answering, doing away with RNNs and replacing them entirely with self-attention and convolution. This model addresses the problem of slow training and inference due to the sequential nature of recurrent neural networks. Similar to the network architecture of BiDFA [1], QANet consists of the embedding layer, the embedding encoder layer, the context attention layer, the model encoder layer and the output layer.

We begin with obtaining context and question embeddings by concatenating pre-trained GloVe and character embeddings, following the Highway module. Instead of recurrent networks as building blocks of the embedding encoder layer, we carried out depthwise separable convolutions [9] and Multi-Head-Attention modules [3]. Having obtained the input embeddings of the context paragraph and the query, the bidirectional attention is computed as described in BiDAF. Similar to embedding encoder layer, the model encoder layer is replaced by the new adaptations of depthwise separable convolutions and Multi-Head-Attention modules. The layer parameters are the same as the embedding encoder layer except that convolution layer number is 2 within a block and the total number of blocks are 7. There are 3 repetitions of the model encoder layer, and the output from 3 repetitions are denoted as M_0, M_1 and M_2 respectively. The output layer predicts the probability of each position in the context paragraph being the start or the end of the answer span. The start probability P_s, end probability P_e and loss function $L(\theta)$ are computed as:

$$\begin{aligned} P^s &= \text{softmax}\left(W_1[M_0; M_1]\right), \quad P^e = \text{softmax}\left(W_2[M_0; M_2]\right) \\ L(\theta) &= -\frac{1}{N}\sum_{i}^{N}\left[\log(P^s_{y^i_s}) + \log(P^e_{y^i_e})\right] \end{aligned} \tag{3}$$

Unified Output Layer Given the base architecture of the QANet and multiple literature reviews, we proposed a few areas of improvements and enhancements. As mentioned in the previous section, the start and end probabilities are computed separately, taking concatenation of outputs from specific model encoder blocks. We have modified such architecture such that the start and end probabilities are modelled using unified weights learning: two output branches were unified into one single branch, and two softmax operations on the first and second channels of the output from a linear layer. In the original architecture, only the first and second encoder blocks are relevant in predicting start probability while the first and third encoder blocks are relevant in predicting end probability. This leads to second encoder block not optimized to predict end probability, and further affecting the optimization of the third encoder block since it takes as input from the second encoder block. Also, we argue finding start and end positions should not be carried out sequentially, as start and end should mutually depend on one another. Thus, we propose to learn both with equal importance. With the reduction of separate branches and elimination of third model encoder block, we hypothesize that this will encourage the model to focus optimizing the parameters in the two remaining model encoder

blocks that can learn both the start and end probabilities with unified branch. The modified model architecture is shown in Figure. 3

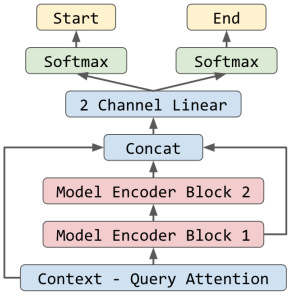

Figure 3: Unified Output Layer

Multitask Learning Based on the previous alteration to the QANet architecture, we further optimized the training process by incorporating an additional task: predicting whether there is answer or not to the question given the context paragraph. Instead of learning it as separate sketchy reading modules described in [4], we formulated it as another learning task with shared underlying network, and added another branch to predict whether there is an answer as shown in Figure. 4.

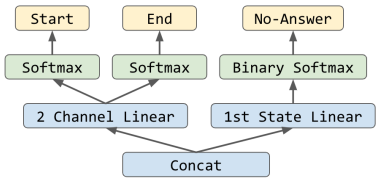

Figure 4: Multitask Learning

Given that a no-answer token has been appended to the beginning of each context, we could derive no-answer information from the 1^{st} hidden state in sequence. So we computed the probability of no-answer by taking the 1^{st} hidden state value from the concatenation of model-layer outputs. We applied a linear function with output size of 2 and a softmax function to compute no-answer probability.

Given the no-answer probability, we update the loss function, where y_s^i and y_e^i are the groundtruth starting and ending position of the i^{th} example, y_n^i is the groundtruth of having answer or not. α_1 and α_2 are hyperparameters that determines the contribution of each task to training:

$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}\left[\alpha_1(log(P^s_{y^i_s}) + log(P^e_{y^i_e})) + \alpha_2 \underline{log(P^n_{y^i_n})}\right] \tag{4}$$

At inference and evaluation, given output start and end probabilities s and e, and the no-answer probability n, we calculate the final no-answer score:

$$\text{score}_{\text{no-answer}} = \lambda_1(s_0 \cdot e_0) + \lambda_2 n_0 \tag{5}$$

4 Experiments

Data. To carry out experiments using the above proposed approaches, we trained and evaluated on SQuAD 2.0 [10], a reading comprehension dataset. The data set consists of paragraphs, questions and answers, and the goal is to infer the answer to the question based on the model's understanding of the context paragraph. Paragraphs in SQuAD are from Wikipedia. SQuAD2.0 combines the 100,000 questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by

crowdworkers to look similar to answerable ones. There are around 150k questions in total, and roughly half of the questions cannot be answered using the provided paragraph. If the question can be answered, the answer will be a chunk of words picked out directly from the context paragraph. This implies implies that the answer does not have to be generated, rather select the start and end index to indicate the span of text. The following example illustrates a typical example from the dataset:

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke).
Answer: not having a residence perm

Before training the model, we have conducted preprocessing work that includes: truncating context and questions with 400 and 50 words respectively, then pad with PAD token. Out-of-vocabulary words are represented by the UNK token. Collecting words representations with pre-trained GloVe word-level embeddings and character-level embeddings. Dividing dataset into train (129941), dev (6078) and test (5915) examples.

Evaluation method As standard metrics for the SQuAD Challenge, we used Exact Match (EM) and F1 as the evaluation metrics. *Exact Match* is a binary measure of whether the system output matches the ground truth answer exactly. *F1* is the harmonic mean of precision and recall:

$$\text{F1} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Experimental details In this section, we discuss the details of our experiments and the evaluated performance of our models on the SQuAD 2.0 dataset. We experimented with various models, each incrementally introduces more components to the baseline model. In particular, we first conducted experiments using the baseline BiDAF, then we added character level embeddings, self attention modules, and gated residual block . We also implemented QANet, followed by our variant of QANet based on gated residual block, unifying optimization of start and end predicting into one module, and multitask learning on no-answer task. To conclude, we developed and evaluated 7 various models:

1. Baseline BiDAF
2. BiDAF + character embeddings
3. BiDAF + character embeddings + self attention
4. BiDAF + character embeddings + gated residual attention
5. QANet
6. QANet + gated residual attention + unified output
7. QANet + gated residual attention + unified output + multitask learning

Before hyper-parameters were finalized, we conducted fine tuning while observing accuracy performances of these models. We have experimented various learning rates ranging from 1e-3 to 0.4, batch sizes ranging from 24 - 64, weight decay rates from 0 - 1e-6, optimizers among ('Adadelta', 'Adam', 'Adagrad', 'SGD'), dropout probabilities ranging from 0 - 0.3 and hidden layer sizes ranging from 64 to 256. We carried out analysis on learning curves and observed the variations in F1, EM and NLL losses, before fixing these parameters for the most optimal experimental run as detailed below.

For experiments on the BiDAF and its variants: we used a ema decay rate of 0.999, learning rate of 0.5, L2 weight decay of 0.0, batch size of 64, encoder hidden layer size of 100, dropout probability of 0.2, Adadelta optimizer, and number of training epochs of 30.

For experiments on the QANet and its variants: we used a ema decay rate of 0.999, learning rate of 0.4, L2 weight decay of 4e-7, batch size of 32, encoder hidden layer size of 128, dropout probability of 0.1, Adadelta optimizer, and number of training epochs of 30-50. We have decreased the batch size to 32 for QANet experiments due to GPU memory limitation. Additionally, for the encoder blocks within the QANet architecture, we used 8 heads in multi-headed self attention, base depth dropout rate of 0.9, 4 Convnet layers, and a hidden size of 128. The modeling encoder stack has 5 layers instead of 7.

Results Using the above proposed hyper-parameters settings, we have initiated multiple experiments using different model architectures. We have obtained the following results shown in Table.1. The best one is also available on the **non-PCE** leaderboard.

	EM	**F1**
Baseline BiDAF	57.89	61.27
BiDAF + Char Emb	61.03	64.62
BiDAF + Char Emb + Self Att	61.87	64.99
BiDAF + Char Emb + Gated Residual Att	62.44	65.88
QANet	62.10	65.10
QANet + Gated Residual Att + Unified Output	63.00	67.16
QANet + Gated Residual Att + Unified Output + Multitask	64.61	68.67

Table 1: Experimental Results

From the BiDAF experiment results, we observe that character-level embedding made significant contribution to F1 and EM scores by considering out-of-vocabulary words and learning structure information. The addition of self attention together with gated residual block has further improved performance of BiDAF. With QANet, we have achieved a further improvement with the novel architecture that incorporates self attention and convolution to eliminate memory-loss in long texts. As shown in Table.1, basic QANet outperformed the BiDAF. With the addition to gated residual module and unification in output layers, QANet has shown further improvements. In addition, the multitask learning with no-answer branch further improve the accuracy. In the process of training, we have also monitored the changes of F1 and EM scores, as shown in Figure. 5 and Figure. 6. These figures illustrate the learning process of various models, and inform saturation.

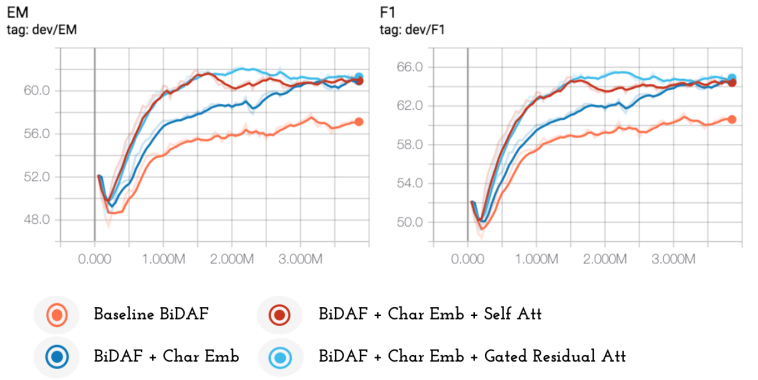

Figure 5: BiDAF: F1 and EM vs Steps

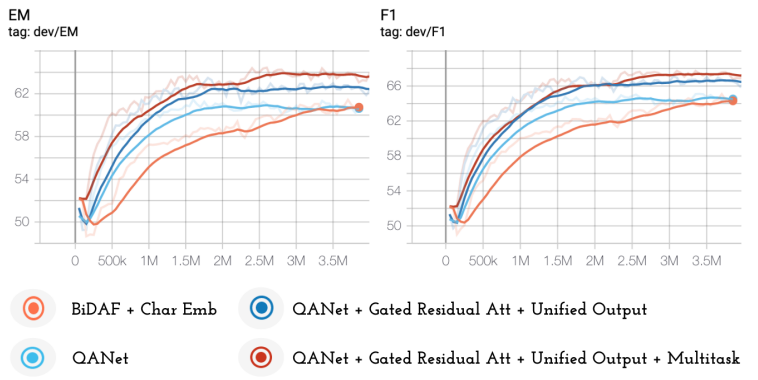

Figure 6: QANet: F1 and EM vs Steps

5 Analysis

Given final dev EM and F1 scores for models outlined in Table. 1 and the learning curves shown in Figure. 5 and 6, we have successfully validated our hypothesis that character embeddings, self attention, gated residual attention, output branch unification and multitask learning had improved baseline state-of-the-art model architectures at different degrees.

To begin, we analyze the results obtained based on BiDAF. With the addition of character embeddings to encapsulate structural compositions of words, the model made use of low-level features of various granularity. In addition, it enabled the model learn out-of-vocabulary words. With a more robust feature extraction pipeline, we were able to serve not only semantic features, but structure features in question-answering models. Referring to Figure. 5, the model with character embedding was able to achieve the same scores with only one third number of steps.

Furthermore, with the addition of self attention module from the Transformer architecture, our model pay attention to information outside of context windows. This addition to the BiDAF attention was proved to be essential, as shown in Figure. 5. By paying a close look at the gradient of the learning curve before 1M steps into training, we can clearly see that self attention encourages efficient learning, with the steepest gradient initially. Lastly, with gated residual blocks, we were able to alleviate the problem of vanishing gradient, and memorize partial features from previous states. The idea of gated residual adopted in our model serves a similar purpose as that in residual neural networks, allowing gradients to take a shorted path backwards. Also, we observed that model learns faster and saturation occurs earlier with the .

Also, we analyze the experiments on top of QANet and validated our hypothesises. Given the advantage of gated residual attention mechanism, we applied it on QANet multi-headed self attention. Previously, the output prediction were branched out to predict start and end probabilities independently of each other. We propose to unify the two output branches, to model mutual dependencies between the model's inference on start and end index probabilities. We argue finding start and end positions should not be carried out sequentially, as start and end should mutually depend on one another. Thus, we propose to learn both with equal importance. With unified output layer, the model learnt to be more robust and can be generalized better, and we were able to remove one model encoder block without causing a decrease in performance and decrease training time.

In addition, we designed a multi-tasking learning algorithm with no-answer prediction. With the modified loss function and inference logic, we observe a further improvement in performance - resulting in our best score for submission to leaderboard. Based on previous experiments, we observe that at early stage of training, the NLL is drastically decreasing due to model always predicting no-answer. By introducing multi-task, we enforced penalty on no-answer misclassification loss, and improved no-answer correctness with modified inference logic. As shown in Figure. 6, with multitask, the problem of sudden drop in EM / F1 at beginning of training was mitigated. Multi-task learning were widely used in many problems, and we have introduced a lightweight task to predict whether there is an answer or not. It helps the learnt parameters to become more generalize to not only question answering tasks but classification task.

6 Conclusion

In this paper, we investigated current state-of-the-art models for question answering tasks and discussed various model architectures and techniques adopted by these models. We carry out experiments using the SQuAD 2.0 dataset and evaluated our models based on standard metrics. We implemented these baseline models, and incorporated our improvements based on our analysis of their shortcomings. We incorporated self attention and gated residual block on top of the BiDAF architecture. We investigated the QANet architecture to verify the feasibility of removing recurrent networks for time-efficient training process. We further improved QANet by re-modelling the architecture such that model learns optimizing both start and end probabilities through unified output layer. Also, we formulated a more advanced loss function and an inference logic that optimizes no-answer classification. We carried out fine-tuning of hyper-paramets and conducted thorough analysis based on learning curves and performance metrics.

References

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

[2] 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. *arXiv preprint arXiv:1804.09541*, 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[4] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *arXiv preprint arXiv:2001.09694*, 2020.

[5] Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced mnemonic reader for machine reading comprehension. *arXiv preprint arXiv:1705.02798*, 2017.

[6] Jeffrey Pennington, Richard Socher, and Christopher D 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.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[10] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.