Transformer-Based Model for Question and Answering

Jennifer She[*]
Department of Computer Science
Stanford University
Stanford, CA 94305

Emma Chen[†]
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

We study the performance of attention-based models in solving the SQuAD 2.0 Question and Answering (QA) challenge, inspired by the Transformer and QANet. We reproduce the QANet model and experiment with variants of QANet. Specifically, we vary the model's components (embeddings, convolutional layers) and incorporating recurrence into to the model, in addition to varying hyperparameters (optimizer, batch size, number of layers, hidden dimension). We find that using regular convolutional layers in place of depthwise separable layers in the model leads to worse training dynamics (with the training loss plateauing at a higher value); adding character embedding improves model performance for both the baseline recurrence-based BiDAF model and QANet; making word embeddings trainable leads to overfitting; and incorporating recurrence into QANet leads to relatively on-par performance.

1 Introduction

The Transformer model [Vaswani et al., 2017] has very recently become the foundation of many advanced, state-of-the-art models in natural language processing (NLP). The purpose of Transformer is to remove recurrent components of a model in order to allow parallelization for faster computation. Inspired by this, we re-implement QANet [Yu et al., 2018], a model based off of the self-attention mechanisms in Transformer, for tackling the SQuAD 2.0 Question and Answering (QA) challenge. Our re-implementation closely mirrors that of Yu et al. [2018] and Chute [2018], as we find that implementation details significantly affect outcome. With this model, we achieve an EM score of 61.2 and F1 score 65.1 on the dev set leaderboard.

Additionally, we explore different variations of QANet to learn about how modifications of embeddings, convolutional layers, hyperparameters and the addition of recurrence (inspired by Transformer-XL [Dai et al., 2018]) affect training outcome. We find character embeddings is relatively important and also improves the performance of the baseline BiDAF model, achieving an EM score of 61.4 and F1 score of 64.9 on the dev set leaderboard. We find that making the word embeddings trainable in the QANet leads to sigificant overfitting, with loss on the dev set increasing sigificantly as the loss on the train set decreases. We find that the regular convolutional layers do not work as well compared to depthwise separable convolutional layers, potentially because the hyperparameters are not tuned to this setting, and the additional parameters are difficult to learn. We also notice that in terms of hyperparameters, having a larger model encoder block (with more layers) results in better performance than having a larger batch size. Finally, we notice that incorporating recurrence in the QANet model not only increases the training time, but also lowers the model performance.

[*] jenshe@stanford.edu
[†] ychen421@stanford.edu

2 Related Work

Related work include both general language models (Tranformer, Transformer-XL) and question-and-answering task specific models (BiDAF, QANet).

BiDAF
The BiDAF model [Seo et al., 2016] is the default baseline for our question and answering task. It is built fundamentally on recurrence, using bidirectional LSTM's for encoding the inputs in the embedding encoding layer, refining their dependencies in the model encoding layer, and transforming hidden representations into the output in the output layer. As an initial task, we build upon the baseline implementation of this model by adding character embeddings.

Despite the popularity of recurrent neural networks, they are inherently sequential, which hinders parallelization. Additionally, the number of operations that are needed to relate two input or output positions scales linearly in the distance between the positions. As a result, higher computation will be needed for larger inputs, and dependencies between distant pairs of positions are harder to capture.

Transformer
The Transformer model [Vaswani et al., 2017] provides an alternative to recurrence for tasks like language modelling and machine translation by using self-attention at its core. The model heavily makes use of "multi-head attention", which are stacks of self-attention modules, learned in parallel and merged together using concatenation and linear operations. These components are incorporated into the encoder and decoder along with feed-forward linear layers, which makes up the model. These self-attention components allow for modelling of direct dependencies between positions within a constant number of steps. However, one potential downside is that in practice, the size of the input (sequence size) must be fixed in practice, in order to eliminate recurrence.

Transformer-XL
Transformer-XL [Dai et al., 2018] builds upon Transformer by reincorporating recurrence in order to allow for longer dependencies beyond fix-sized inputs. The model breaks up long sequences into segments, and learns this segments sequentially, using hidden states of previous segments as additional inputs for learning later segments.

QANet
QANet [Yu et al., 2018] applies the self-attention mechanism from Transformer, with the addition of depthwise separable convolutional layers, to the question and answering task. Thus, QANet eliminates the recurrent neural network structure of BiDAF. Inspired by the Transformer model, we reproduce this model, modify its various components, and build upon it using recurrence concepts from Transformer-XL.

3 Approach

Baseline
Our baseline is the standard SQuAD BiDAF model, without character embedding.

Character Embedding
We add character embedding to the baseline BiDAF model, and the QANet model. We retrieve the raw character embeddings of each character of each word in a given sequence. This is fed into a dropout layer, a 1D-convolutional layer with a kernel size of 5 as well as a reLU layer. We then take the maximum value in the character dimension of each word and concatenate this to the initial word embedding. We set the number of character embeddings to be 1400, which covers all possible characters in SQuAD 2.0 and set the embedding size of each character to be 200.

QANet model
For the transformer-based model, our approach is based off of QANet [Yu et al., 2018], and we extend this model in the final report. To implement the QANet model, we made modifications to the

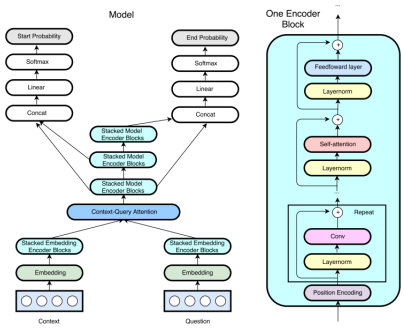


Figure 1: QANet Architecture [Yu et al., 2018]

BiDAF baseline by adding positional encoding, and modifying the encoder architecture as well as the output layers, using existing code for positional encoding and multi-head self-attention from the Transformer written in Pytorch [Huang, 2018]. Our specific architecture is shown in figure 1.

We reuse the embedding layer (with the additional character embeddings) and the context-query attention layer from the provided BiDAF model.

Both the embedding encoder and model encoder layers are composed of stacks of encoder blocks, as shown in the right side of figure 1. Before these blocks, we apply a linear projection layer to ensure that that the input has the correct hidden dimension size for residual connections. Within an encoder block, positional encoding is added to the input, and then passed through some number of depthwise separable convolutional layers, a single self-attention layer and a feedforward layer. We additionally have layer norm (which normalizes over the sequence dimension) before each layer, and dropout and residual connections after each layer. Following the setting of QANet, we using 1 encoder block with 4 convolutional layers in the block, and a convolutional filter size of 5 for the embedding encoder layer, and we use 3 repetitions of 7 encoder blocks with 2 convolutional layers in each block and a convolutional filter size of 7 for the model encoding layer. For the self-attention layers, we use 4 heads instead of 8 in order to conserve memory, as per Chute [2018]. The output from the first and second repetitions of the model encoding blocks are fed into the output layer for determining the starting position, and the outputs from the first and third repetitions are used for determining the ending position.

In the output layer, we remove the LSTM from the BiDAF output layer, and simply keep the linear and sigmoid layers.

Depthwise separable convolution
Depthwise separable convolutions are used in every encoder block instead of normal convolutions, which makes a huge difference because it largely reduces the number of parameters and multiplications needed, while maintaining good performance. This allows us to keep a descent batch size and model encoder blocks size when there is limited memory and computation resource. The key idea of separable convolution is to replace a big kernel of size $\text{len}_k \times \text{len}_k \times \text{channel}_{\text{input}}$ with several smaller kernels. In the case of depthwise separable convolution, the original kernel is divided into smaller kernels along the feature dimension, with each smaller kernel having a size of $\text{len}_k \times \text{len}_k \times 1$. After calculating the convolution with the smaller kernels, the intermediate output, now with a size of $\text{len}_k^* \times \text{len}_k^* \times \text{channel}_{\text{input}}$, will convolve with $\text{channel}_{\text{input}} \cdot \text{channel}_{\text{output}}$ 1 x 1 kernel, yielding an output of size $\text{len}_k^* \times \text{len}_k^* \times \text{channel}_{\text{output}}$.

Trainable Word Embeddings - Original
We notice that the default setting freezes the word embeddings to be the pretrained word vectors throughout training. We experiment with allowing the word embeddings to be trainable, with hopes that the word embeddings will be more adapted to the task.

Incorporating Recurrence - Original
Noticing that the QANet model tends to predict positions near question keywords in the context (see examples section below), as well as being inspired by Transformer-XL and its ability to handle variable-length inputs, we incorporate recurrence into the original QANet model. Instead of feeding the whole context into the model encoding layer at once, we divide it into segments and feed them

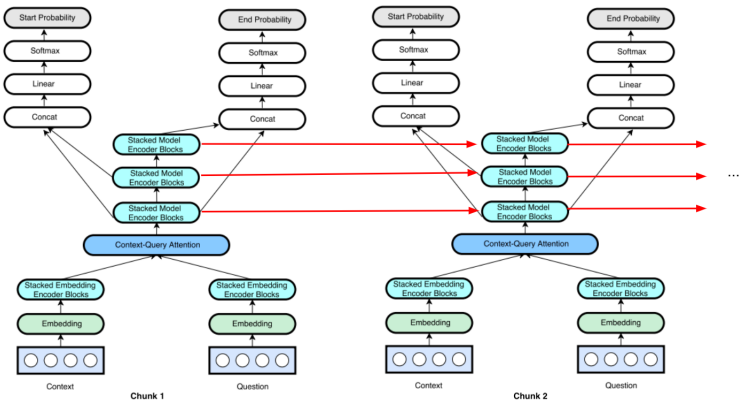


Figure 2: Recurrent QANet (Modified from Yu et al. [2018])

into the encoder one at a time. The model encoder blocks' output of the previous segment is saved and added to the next segment as input into the model encoder blocks at the next timestep, as illustrated in figure 2. Since there is no previous segment for segment 1, nothing is added to the first segment. This way we can train on very long examples with a fixed segment size. We also think it will strengthening the dendency between context positions that are further apart in distance.

Hyperparameters tuning
Additionally, we experimented with tuning hyperparameters (batch size, char embedding size, number of layers in model encoder layer) and other components, including the optimizer and learning rate.

4 Experiments

4.1 Data

We use the SQuAD 2.0 dataset provided by the course with custom dev and test set. More details on the dataset is provided in the final project handout.

4.2 Evaluation method

We use Exact Match (EM) and F1 for metrics to measure the model's performance. On the leaderboard of SQuAD 2.0, the EM score of models that contain BiDAF ranges from 59.2 to 68.0, while the F1 score ranges from 62.1 to 71.6. QANet is not used by top models on the SQuAD 2.0 leaderboard, but the best ensemble QANet model ranks third on the SQuAD 1.0 leaderboard.

4.3 Experimental details

Our experiments include the default BiDAF model with the addition of character embeddings, the re-produced QANet model, and variants of the QANet model. All of them are trained for 30 epochs or until they no longer show improvement.

Model #1: BiDAF + Char Embedding
We use the configurations provided by the default code (AdaDelta optimizer, learning rate set to 0.5, hidden dimension set to 100, and batch size set to 64). Additionally, the size of character embeddings is set to 200.

Model #2: QANet + Char Embedding
We modify the BiDAF configurations in order to allow better training of the QANet. Specifically, we use the ADAM optimizer, with learning rate set to 0.001, β_1 set to 0.8, hidden dimension set to 96 and batch size set to 16. We use a warm-up learning rate that exponentially increases from 0 for the first 2000 iterations (modified to compensate for the smaller batch size).

Model #3: QANet (Conv Layers) + Char Embedding
We use all of the same settings as model #2, with the exception that the depthwise separable layers are replaced by regular convolutional layers (with the same kernel size of 5).

Model #4: QANet (Trainable Word Embedding) + Char Embedding
We use all of the same settings as model #2, with the exception that the word embeddings are trainable.

Model #5: QANet (Recurrence) + Char Embedding
We use all of the same settings as model #2. Additionally, the inputs into the modelling layer are divided (over the sequence dimension) into 2 or 4 segments.

4.4 Results

Below is the performance of our models on the non-PCE dev leaderboard.

Model	F1	EM
#1 BiDAF + Char Embedding	64.9	61.4
#2 QANet + Char Embedding	65.1	61.2
#3 QANet (Conv Layers) + Char Embedding	51.8	51.6
#4 QANet (Trainable Word Embedding) + Char Embedding	54.8	52.3
#5 QANet (Recurrence) + Char Embedding (2 segments)	60.2	56.6
#5 QANet (Recurrence) + Char Embedding (4 segments)	61.5	57.8

Table 1: Model Scores

From table 1 we can see that QANet + Char Embedding has higher F1 than BiDAF + Char Embedding, but lower EM. Comparing model 2 and 3, we see that replacing depthwise separable convolution with normal convolution will drastically lower the model performance. Comparing model 2 with 4 we see that making the word embeddings trainable also reduces model performance. Comparing model 5 with 2 segments and 4 segments, we see that a smaller segment size tends to give better performance.

Additionally, we submitted the QANet + Character Embedding model to the non-PCE test leaderboard, and achieved F1 and EM of 61.7 and 58.2.

5 Analysis

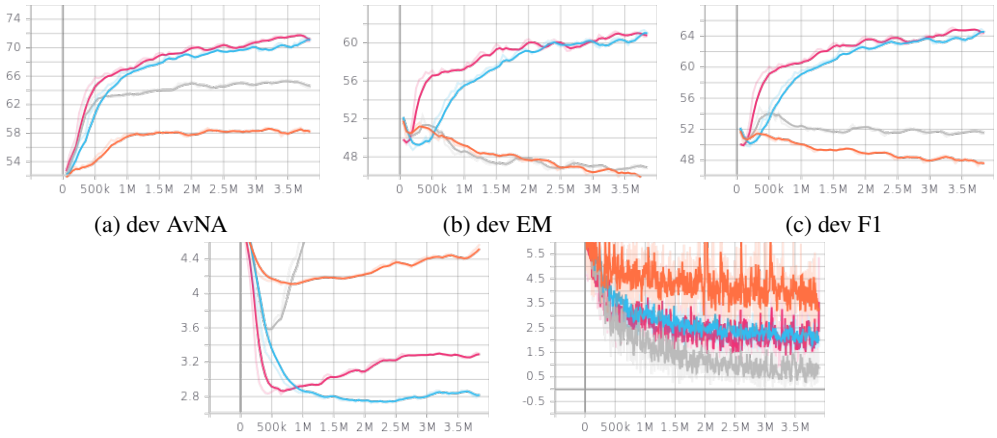
5.1 Training Dynamics

A comparison of the training and evaluation metrics over model training are shown in figures 3 and figures 5.

Character Embedding
We notice that the training dynamics of the BiDAF model with the addition of character embeddings is reasonable. Both F1 and EM initially dip in the beginning of training, but increases with further decrease of the NLL objective.

QANet
Compared to the BiDAF model, the initial dip in F1 and EM scores during training before its eventual increase is much smaller, meaning that QANet is likely able to train faster in the initial stage. However, we notice that although NLL decreases on the train set, it eventually increases slightly on the dev set. This means that the model likely overfits slightly, even though F1, EM and AvNA scores all monotonically increase throughout training. Additionally, we ran the same setting several times, and notice that some runs of the experiment have better training dynamics than others (see figure 4a). One potential explanation for this is local minima.

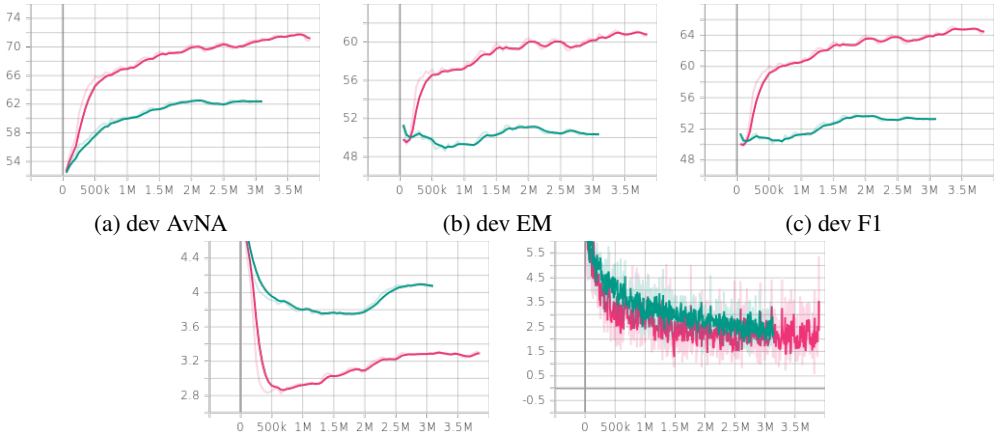
Normal Convolutional Layer
We notice that when we replace the depthwise separable convolutions with regular convolutions, the NLL scores of both the train set and the dev set plateau at a much higher value of approximately 4,



(a) dev AvNA (b) dev EM (c) dev F1

(d) dev NLL (e) train NLL

Figure 3: Training Dynamics: (blue) BiDAF + Char Embedding, (pink) QANet + Char Embedding, (orange) QANet (Conv Layers) + Char Embedding and (grey) QANet (Trainable Word Embedding) + Char Embedding.



(a) dev AvNA (b) dev EM (c) dev F1

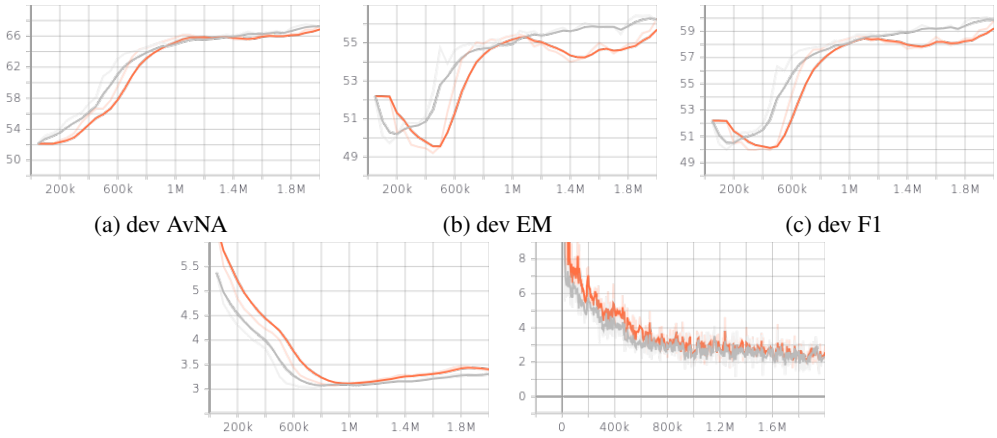
(d) dev NLL (e) train NLL

Figure 4: Training Dynamics: 2 different runs of QANet + Char Embedding

compared to 2.5 in the default setting. One possible explanation is that the training setting (optimizer, learning rates), which are tuned for the original QANet model, does not work well for this new model with normal convolution. This new model likely needs smaller learning rates in order to avoid instability, due to its higher number of parameters. As a result, in our setting, the model is not able to learn effectively in order to decrease its objective beyond a certain threshold. Another explanation is that the additional parameters are unnecessary, and impedes learning because the pattern that is shared among different embedding layers, which can be captured easily by depthwise separable convolutions, cannot be captured easily by regular convolutions.

Trainable Word Embedding

We notice that while this model achieves a NLL score on the training set compared to the QANet model, the NLL score on the dev set increases much more significantly. As a result, both F1 and EM decreases over time. This is likely because making word embeddings trainable allows the model to overfit too much. One possible explanation for this is that the train set is too small compared to the dataset that is used to learn the word vectors, and as a result, the learned word embeddings is no longer generalizable to the dev and train set.



(a) dev AvNA (b) dev EM (c) dev F1

(d) dev NLL (e) train NLL

Figure 5: Training Dynamics: Recurrent QANet with 2 segments (orange) and 4 segments (grey)

Recurrence
Both recurrence models perform decently well, although they still perform worse than QANet + Char Embedding alone. We think this is because the hyperparameters used are tuned for the QANet + Char Embedding model, which may not be the most suitable set of hyperparameters for the recurrence model. It is also possible that this model is not a good approach to strengthen dependency of neighboring context. We are also aware that dividing into 2 or 4 segments may not be enough but more and smaller segments may help the model performance. Finally, another disadvantage of this model is longer training time due to sequential nature of recurrence.

Hyperparameters
For variations of the QANet, we also notice that in terms of the trade off between batch size and model encoder block size (the number of encoder blocks within a model encoder), having a larger model encoder block produces better score than having a larger batch size.

5.2 Examples

We examine some sample inputs and outputs from our best performing QANet model. We notice that the model often fail to figure out whether a question has answer or not. In particular, we notice a trend that the model tends to fault on predicting answers to questions that have no answer, especially "when" and "what" questions that appear easy.

```
Question: What is the highest reference hospital in all of
    Germany?
Context: Today, Warsaw has some of the best medical facilities
    in Poland and East-Central Europe. The city is home to the
    Children's Memorial Health Institute (CMHI), the highest-
    reference hospital in all of Poland, as well as an active
    research and education center. While the Maria Sk odowska-
    Curie Institute of Oncology it is one of the largest and
    most modern oncological institutions in Europe. The
    clinical section is located in a 10-floor building with 700
     beds, 10 operating theatres, an intensive care unit,
    several diagnostic departments as well as an outpatient
    clinic. The infrastructure has developed a lot over the
    past years.
Answer: N/A
Prediction: Children's Memorial Health Institute

Question: When did King Harold II conquer England?
Context: In 1066, Duke William II of Normandy conquered England
     killing King Harold II at the Battle of Hastings. The
```

```
    invading Normans and their descendants replaced the Anglo-
    Saxons as the ruling class of England. The nobility of
    England were part of a single Normans culture and many had
    lands on both sides of the channel. Early Norman kings of
    England, as Dukes of Normandy, owed homage to the King of
    France for their land on the continent. They considered
    England to be their most important holding (it brought with
     it the title of  K i n g an  important status symbol).
Answer: N/A
Prediction: 1066
```

We think QANet alone may not be good at distinguishing whether there is answer or not. One thing we can do in the future is to train a separate model to classify whether a question has answers, and then use QANet to find the answer if there is one. Alternatively we can look into an additional answer verification process, which will look at the output of the QANet model in order to re-evaluate answerability of the question.

We also notice that the model tends to choose neighboring texts of the question keywords that appear in the context as the answer. Below is a typical example of such a case: "OAPEC ... block oil deliveries to the United" shows up in the last sentence of the context. The model predicts "principal hostile country", which shows up in the same sentence with the above keywords, while the correct answer should be "American aid to Israel", which is at the very beginning of the context. This pattern suggests that incorporating recurrence similar to what we did as an extension of the QANet, might be a good future direction.

```
Question: Why did OPEC block oil deliveries to the United
    States?
Context: In response to American aid to Israel, on October 16,
    1973, OPEC raised the posted price of oil by 70%, to $5.11
    a barrel. The following day, oil ministers agreed to the
    embargo, a cut in production by five percent from September
    's output and to continue to cut production in five percent
     monthly increments until their economic and political
    objectives were met. On October 19, Nixon requested
    Congress to appropriate $2.2 billion in emergency aid to
    Israel, including $1.5 billion in outright grants. George
    Lenczowski notes, "Military supplies did not exhaust Nixon'
    s eagerness to prevent Israel's collapse...This [$2.2
    billion] decision triggered a collective OPEC response."
    Libya immediately announced it would embargo oil shipments
    to the United States. Saudi Arabia and the other Arab oil-
    producing states joined the embargo on October 20, 1973. At
     their Kuwait meeting, OAPEC proclaimed the embargo that
    curbed exports to various countries and blocked all oil
    deliveries to the US as a "principal hostile country".
Answer: American aid to Israel
Prediction: principal hostile country
```

6 Conclusion

In summary, we achieved relatively competitive performance, re-implementing QANet and adding character embeddings to the default BiDAF model. During hyperparameter tuning, we found that under memory constraints, a combination of bigger model encoder size and smaller batch size gives better performance than the opposite. We also experimented with different variant of QANet and learned that depthwise separable convolution gives a key improvement in model performance compared to normal convolution. We experimented with adding recurrence to the QANet, and dividing contexts into smaller segments seem to work slightly better. Future directions include modifications of QANet that can better allow for recurrence (such as relative positional encoding)

and additions to the QANet (answer verification, multiple-step process) that can better handle the answerability of questions.

References

C. Chute. squad-transformer. 2018. `https://github.com/chrischute/squad-transformer`.

Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

Y.-H. Huang. attention-is-all-you-need-pytorch. 2018. `https://github.com/jadore801120/attention-is-all-you-need-pytorch`.

M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.