Closed-domain Question Answering: Extending the BiDAF Architecture

Spyros Barbakos

January 2023

Abstract

Machine comprehension is one of the most important fields in NLP, as it enables machines to understand and process natural language text, which is the primary means of communication between humans. This allows machines to perform a wide range of tasks, such as answering questions, summarizing text, and generating responses, which can be applied in various fields such as customer service, healthcare, and education. In this project, I extend the BiDAF (Bidirectional Attention Flow) model, which is one of the most well-recognized neural network architecture used for closed-domain question answering. I improved the baseline model by implementing learnable character Embeddings and I replaced the RNNs of the baseline model with encoder blocks from the QANet architecture. The final model achieved an **F1 score of 68.35**, whereas the baseline model an **F1 score of 59.5**

1 Introduction

Reading comprehension for AI systems involves understanding a piece of text well enough to answer questions about it. This task requires: accurate modeling of the semantics of each word in the context and question, the ability to keep track of information from different parts of the text and the ability to determine if a question is answerable based on the given context. Traditional machine learning models struggle with this task.

One of the most common ways to evaluate an AI system designed to answer questions is the Stanford Question Answering Dataset (SQUAD 2.0). This Dataset contains triplets, that each one of them consists of a paragraph, a question on this paragraph, and an answer. The answer is always present in the body of the paragraph, so the objective of the model is to identify the precise location of it in the given paragraph.

In the last few years, the field of NLP has seen great advancements, due to the development of the pre-trained language models, which are the most well-performing models in almost every NLP application. So, Pre-trained models have achieved remarkable success on the SQUAD dataset, with F1 scores surpassing 90, outperforming even human performance.

In this project, I explored two neural architectures and their variations: Bidirectional Attention Flow and QANet, which use pre-trained word Embeddings, but are not pre-trained language models.

2 Motivation

The Attention mechanism is the key functionality that makes machine comprehension possible for both models.
The BiDAF model introduced the idea of the attention that flows both ways, i.e., from context to question and from question to context. This bidirectional attention makes the model able to observe how parts of the question interact with parts of the context and vice-versa. The parameters are learned in such a way, that the attention layer outputs a representation that encodes the information given in the context with respect to the question, by highlighting the parts that are more relevant to the query. With the right decoding, this encoded information reveals the span of the answer.

Despite the fact that the BiDAF model uses attention, it relies heavily on Recurrent Neural Networks to encode and decode information that enters and exits the attention layer.

As the field of NLP advances, there is a growing trend towards utilizing pure attention-based mechanisms in place of traditional recurrent neural networks. Research has shown that those mechanisms are superior to RNNs, because they overcome the two major negatives of the RNNs. The one is that because of their sequential nature, their training cannot be parallelized and the second one is that they fail to model long-term dependencies between words, which is known as the vanishing Gradient problem. LSTMs and GRUs nearly fixed this problem, but it is still present to some extent.

Attention is the key component of the transformer architecture, which is becoming the best-performing architecture in all the applications of NLP. One reason why this architecture took off is the fact that it discarded the Recurrent Neural Networks and completely replaced them with modules that use the attention mechanism, which had better performance and whose training could be paralelized in hardware, making it exceptionally fast.

3 Models

3.1 Qanet

Having the above in mind, I implemented a version of the QANet architecture that uses the BiDAF attention, but completely replaces the RNNs with encoder blocks. The model consists of 5 parts: An Embedding layer, an Embedding encoding layer, an attention layer that enables attention to flow from query to context and vice-versa, three model encoding layers that share weights and one output layer.

3.1.1 Embedding Layer:

This layer produces a representation for each word, that consists of the concatenation of the fixed GLoVe vector of this word and its learnable character vector (character vectors are explained below). The concatenation is then passed through a two-way highway network to facilitate information flow across the next layers [4].

3.1.2 Embedding Encoder and Model encoding Layers:

Those layers are built from encoder blocks, which are going to be analyzed in this section.
Intuition: The problem that this layer solves, is that each word-vector coming out of the Embedding layer encodes the meaning of the standalone word. However, the meaning of each word is influenced by its context. The concepts that every text tries to express are conveyed through sentences and not individual words. So, in order for our model to "understand" those concepts and therefore answer question, we need a way to incorporate to every word vector, the meaning of its context. This is exactly what encoder blocks do.

Each encoder block consists of 4 parts. Because they do not use Recurrency, there is no direct way to model the sequence of the input data, so the first part is a positional encoding layer. Through this layer, each representation that enters the block gets enriched with information about its position in the input sequence [3]. Next, the input gets passed into a sequence of Convolutional blocks that aim to " capture the local structure of the text". Then, the input is passed through a self-attention layer that aims to "capture the global interaction between each pair of words" [2]. Finally, the input is passed through a Feed-forward layer.

The Convolutional network uses the technique of Depthwise Separable convolution to speed up the process and the feed forward layer uses ReLU activations [3][9].

The self-attention block uses multi-head attention. In this mechanism, each head is responsible for modeling a different kind of interaction between words. For example, there may exist a head, whose matrices are learned in a way that captures the possessive relationship between two concepts.
Additionally, there exists a layer normalization layer before each block, that allows smoother gradients, faster training, and better generalization accuracy [7]. And finally, each block is connected to the next via a residual connection [3].

3.1.3 Attention Flow layer

This layer was taken as is, from the starter code. So, it will not be explained in detail here, but we will focus on the intuition behind it. The attention layer outputs a big matrix, where each column corresponds to a context word. The parameters of this layer are learned in a way, that each column gets to be a representation of the corresponding word, that is at the same time "aware" of the existence of the Query and has incorporated relevant information from it [5].

That means that that the produced representation has the meaning of weighing the context words that are more relevant to the query.

3.2 Output Layer

the starting position of the answer is determined using the outputs of the first two groups of encoder blocks, while the ending position is determined using the outputs of the first and third groups of encoder blocks. The probability distribution for both positions is computed using a linear and softmax layer. The loss function is the negative logarithm of the actual starting and ending positions, and it is averaged across the dataset.

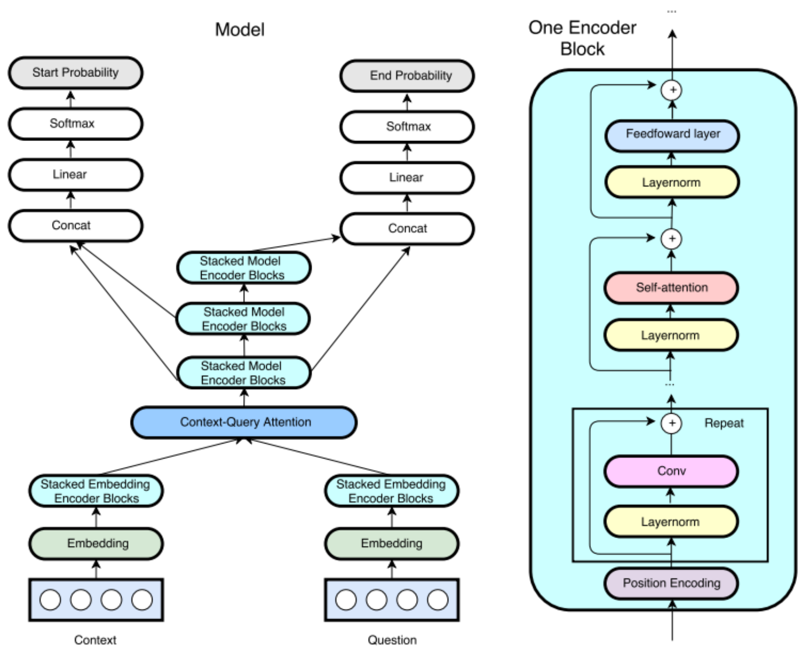


Figure 1: The QANET model

3.3 Baseline

The Baseline model is the BiDAF model, as described in the paper, but without the character Embedding [1]. The code was provided in the handout so I will not explain it further. Here is an illustration of the BiDAF model.

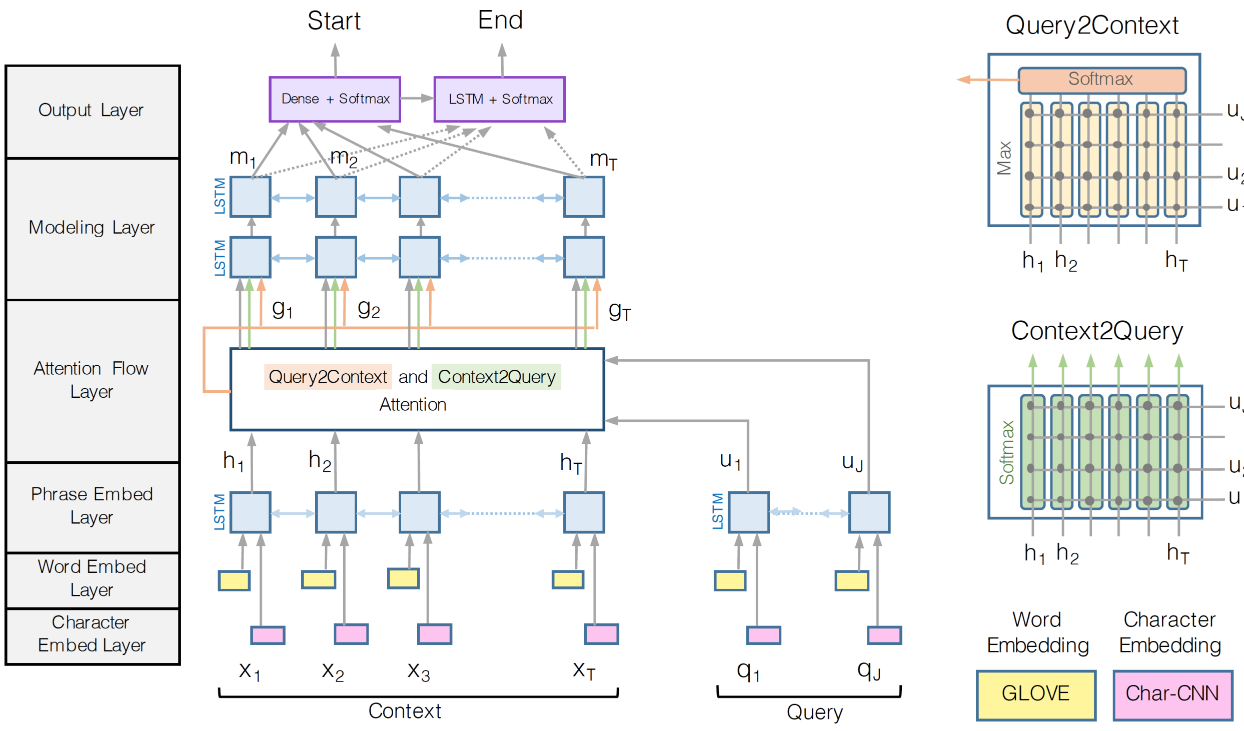


Figure 2: The BiDAF model

4 Implementation

4.1 Character Embedding

The pre-trained GloVe word Embedding give a unique meaning to each words, that is represented as the position of this word in the Embedding space.

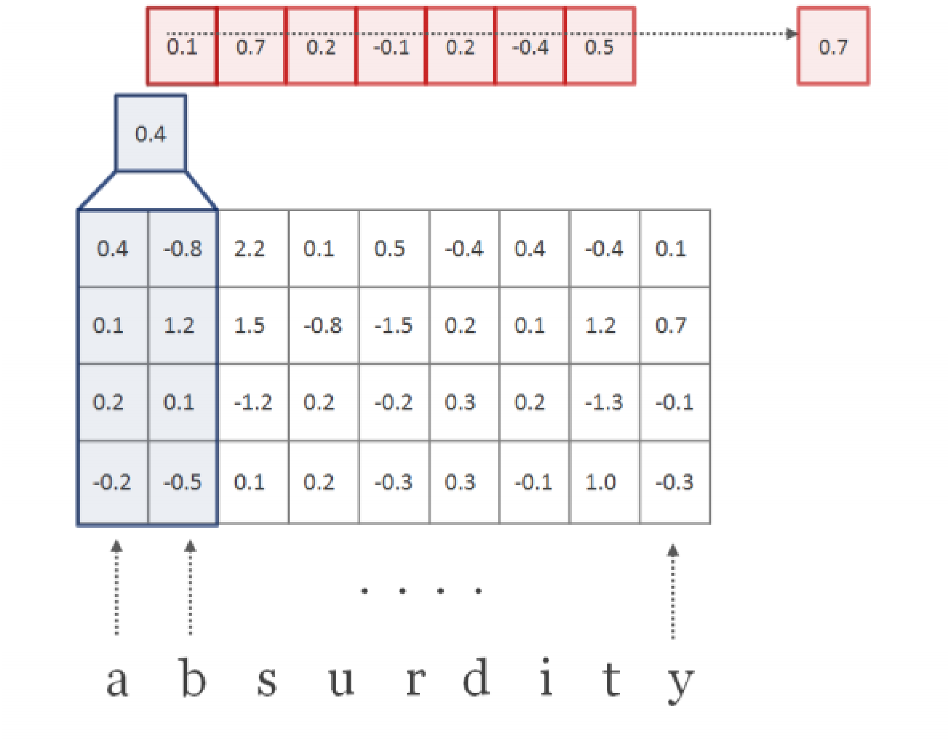
However, words do not only have a meaning as a whole, but individual parts of the word can convey meaning too. For example, we can safely assume that words that use the prefix "mis" have something in common.
Character Embedding have the role of enhancing the word meaning representation by adding information about the meaningful parts of the word. In addition, some words are not present in the GLoVe Embedding and are considered OOV words. The only way that we can add meaningful information to their representation, is through character Embeddings. (Otherwise, it would be randomized)

At the beginning, each word becomes a matrix, where each column corresponds to a character. Those columns are initialized by pre-trained character vectors provided by the starter code.

Next, we apply 1D convolution using several kernels of different sizes. Each kernel slides through the matrix and produces a number on each step. When the kernel has sweeped along the whole word, we have a vector that consists of the numbers produced at each step. We choose the largest of those numbers to represent the kernel. We repeat the same process for all the available kernels. Finally, we stack those numbers on top of each other and we end up with the character Embedding for a specific word.

The kernels are learned in a way that each one of them focuses on capturing a specific subword. In my implementation, I decided to use kernels of size 3. This is because I assumed that the meaningful subwords rarely exceed three letters.

The Illustrated Guide to Bi-Directional Attention Flow (BiDAF) article helped me understand the concept and then I implemented it in Pytorch from scratch. [5]



4.2 Encoder Blocks

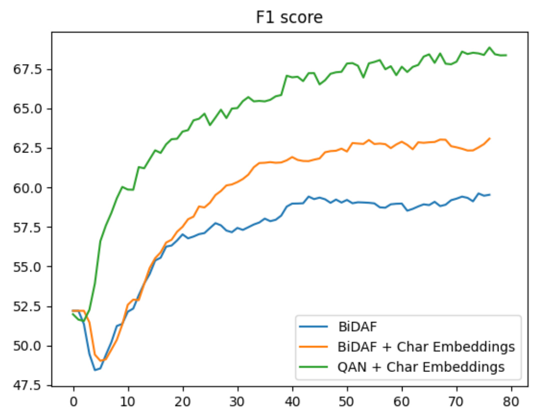
I implemented the encoder blocks from scratch, by reading and understanding the QANET paper (Their functionality is explained above).
For the Embedding encoder layer , I used blocks with 4 stacked convolutional layers and 8 attention heads. For the model encoder layer, I used blocks with 4 stacked convolutional layers and again 8 attention heads. The QANET paper proposes using 7 stacked encoder blocks at each group, but due to GPU memory limitations, I only used 4 block in each group [2].

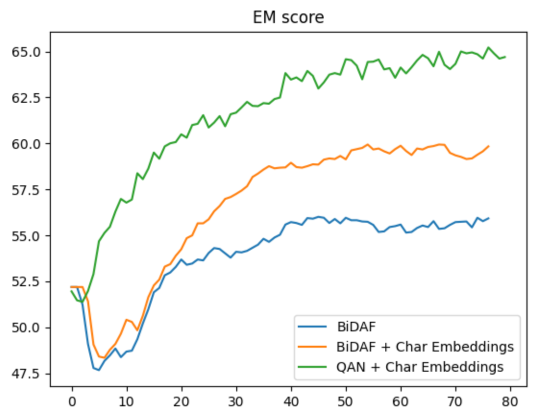
Evaluation method I used the default SQuAD evaluation methods, i.e., the EM and F1 metrics . The EM metric awards a score of 1 for an exact match with the reference answer and 0 for any other answer. The F1 metric, on the other hand, considers precision and recall of predictions, and is calculated as (2 * precision * recall) / (precision + recall). Additionally, the AvNA score indicates the percentage of accurate predictions made on whether or not a question can be answered.

5 Experiments and results

For all the experiments, I used the default pytorch initializations.
Despite the fact that the QANET model avoids recurrency and can be trained in parallel, the training speed was a lot slower. That can be attributed to the fact that teh QANET model is way bigger than the BiDAF model and has more parameters to tune. All the training was done in GPUs.

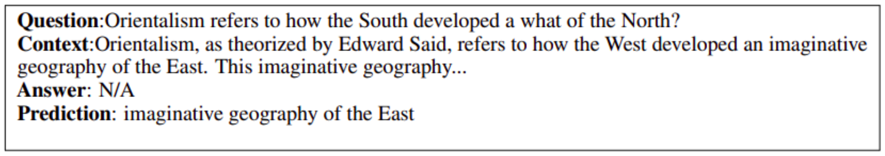
Model	F1	EM
BiDAF (Starter code)	59.53	55.92
BiDAF (Character Embeddings)	63.08	59.84
QANET	68.36	64.7





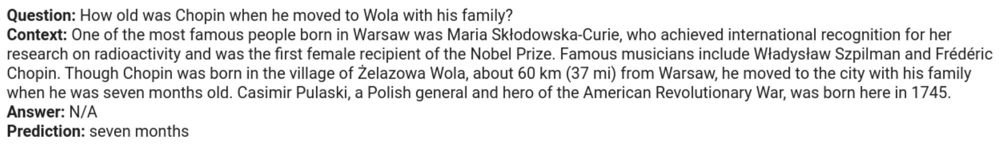
It is obvious that The addition of Character Embeddings improved the performance greatly. Furthermore, the replacement of the RNNs with encoder blocks also improved the performance of the model, for the reasons described above.

6 Analysis



Question:Orientalism refers to how the South developed a what of the North?
Context:Orientalism, as theorized by Edward Said, refers to how the West developed an imaginative geography of the East. This imaginative geography...
Answer: N/A
Prediction: imaginative geography of the East

We can observe that the model produced an answer while it shouldn't, as there is no relevant information in the text. The question asks about the North, while the texts provides infomation about the east. I believe that the reason that the model produced an answer, is that the representations of the words north and east interact heavily, so this answer span is produced. To avoid that, the Embeddings of the words should not be that similar. However, "South" and "East", surely enough, often appear in the same context, so their Embeddings are close. The models described in this Report do not have the ability to solve this problem apparently.



Question: How old was Chopin when he moved to Wola with his family?
Context: One of the most famous people born in Warsaw was Maria Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the Nobel Prize. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.
Answer: N/A
Prediction: seven months

In this passage, we learn that Chopin moved to Warsaw when he was seven months old. However, the question asks how old was he when he moved to Wola (not Warsaw). Hence, the question should have no answer. Despite that, the model produces the answer "seven months". It is obvious that there a lot of parts of the text that align with parts of the question, however the different town name should ideally force the model produce no answer. I believe that this is because names are in fact OOV (out of vocabulary) words and their vector representation is not strong enough to influence the interaction of the context and the Query. This is another weakness of those models.

7 Conclusion

Firstly, we can safely conclude that adding a Character Embedding module greatly improves performance for both models, as it enriches the representation of The Words. Next, we can see that the utilization of multi-head attention increases the performance, as it is able to model different kinds of relationships between words and therefore answer questions more accurately.

Some weaknesses of the models are described in the previous section. It is understandable that these weaknesses could be addressed if the models were designed to incorporate built-in knowledge. For example, they could understand that a certain word is a place name and not just some random OOV word. Also, they could understand the importance of a certain concept, based on their knowledge and enrich its representation accordingly, so that the model will know how to make it interract with the context in order to give better results.

Large Language Models excel in precisely this area, making them an invaluable resource for enhancing the performance of Question Answering models. Incorporating these models can lead to a significant improvement in their overall capabilities.

8 References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi: "Bidirectional Attention Flow for Machine Comprehension", 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. CoRR, abs/1804.09541, 2018.
[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,

Aidan N. Gomez, Lukasz Kaiser: “Attention Is All You Need”, arXiv:1706.03762, 2017
[4] Rupesh Kumar Srivastava, Klaus Greff, Jurgen Schmidhuber: Highway Network, arXiv:1505.00387, 2015
[5] https://towardsdatascience.com/the-definitive-guide-to-bidaf-part-3-attention-92352bbdcb07
[6] 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.
[7] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, arXiv:1607.06450, 2016
[8] A tensorflow implementation of qanet for machine reading comprehension. https://github.com/NLPLearn/QANet
[9] https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728