MuBAF: Multi-Head Bi-Directional Attention Flow For Machine Comprehension

Muhammad Umar Salman
MBZUAI
umar.salman@mbzuai.ac.ae

Abstract

Measuring the machine's ability to understand a text to the extent where it can answer complex questions lies at the root of Natural Language Processing. In recent years, with the growing fame of deep learning and the creation of large-scale datasets we have seen works such as BERT, GPT and XLNET surpassing human performance in numerous NLP tasks such as Question Answering systems. While these transformer-based architectures have managed to achieve state-of-the-art results on high quality Machine Reading Comprehension datasets they have huge disadvantages. These include being compute-intensive and performing poorly on datasets pertaining to specialized fields with domain-specific terms such as medicine and astronomy. In this paper we aim to show why we are moving away from models like BERT to LSTM-based models for production-based Question Answering systems. In this paper we will build upon the BiDAF model by making use of recent innovations such as Multi-Head Attention and LSTM-based contextualized word embeddings.

Keywords: Machine Reading Comprehension, Question Answering, Natural Language Processing

1. Introduction

Machine Reading Comprehension (MRC) is the ability for a system to read a passage and through understanding the underlying meaning, answer a question based on that portion of the text. The machine is either expected to answer one or a set of multiple queries related to the passage. An example of MRC can be visualized in Figure 1. Traditional solutions make use of techniques such as linguistic analysis, pattern matching, syntactic parsing, stemming, semantic parsing, named entity recognition and question classification. Later on more mathematical and statistical methods evolved which analyzed the relationship between the question and answers by calculating the similarity between the question and the sentence of the passage with the most likely answer in it. However, a major drawback of such methods is that they lack the ability to reason and provide answers.

The emergence of deep learning technologies such as sequence-to-sequence, word embeddings and attention mechanism, has led to an improvement in the quality of answers for MRC questions. Most LSTM-based models of previous architectures closely follow the two neural models proposed in [1] which are AR (The Attentive Reader) and the IR (Impatient Reader) which use Bidirectional Long Short-Term Memory (Bi-LSTMs) [2] to encode the corpus and the attention mechanism to map the question to the corresponding text in the passage. In more recent works such as GPT-2 [3], BERT [4] and XLNET [5] we can see the models surpassing human performances on several QA datasets such as SQuAD[6], RACE [7] and CNN/Daily Mail [1] etc.

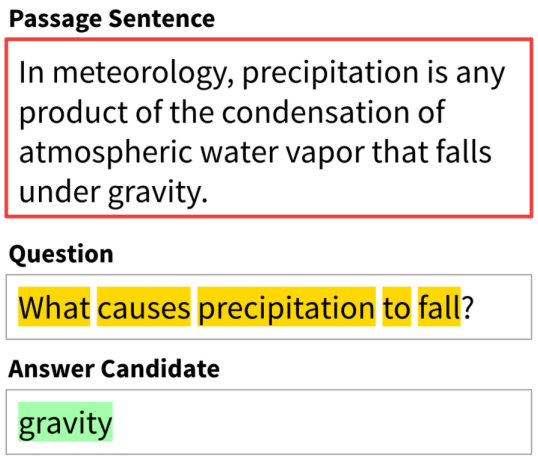

Figure 1. The figure shows an example of MRC where the answer of the question posed is extracted from the text of the passage itself.

The code is available on the following link [1]

[1] ABC

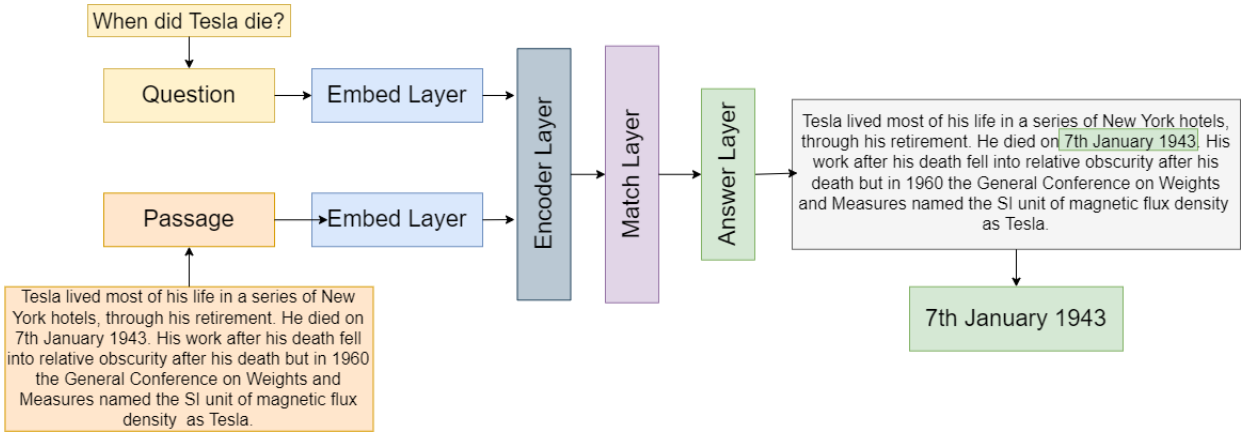

Figure 2. General architecture of MRC based LSTM/RNN QA models.

The following are our contributions while building upon BiDAF[8] :

- We use contextualized word embedding's instead of character-based embeddings which was inspired by (LSTM-based contextual embedding) ELMo[9]. The choice of preference over the two was because contextual embeddings assign each word with a representation based on its context, thereby capturing uses of words across varied contexts.
- Adopt a Multi-Head Attention architecture instead of the already existing single attention layer as multiple heads provide the model with various different aspects of the query-aware representation which captures richer representations.
- Made use of multi-layered Fully-Connected Network before softmax to predict the indices of start and end of the span.
- Lastly, created a cleaned and processed dataset with proper index labels, tokenization and POS and NER Tags .

However, the question lies in why we prefer LSTM/RNN-based model architecture over the state-of-the-art BERT [4]. Models that use BERT for their contextualized embeddings such as SPAN-BERT [10], LUKE [11], XLNET [5] etc have out-performed human performances on various QA datasets. Although these models achieve state-of-the-art results on standard datasets, they still have massive disadvantages. BERT which is used to generate contextualized word embeddings is extremely compute-intensive at inference time which means its use in commercial production would prove very costly. Also the deployment of these models in dynamic commercial environments often yield poor results. One reason for this is because commercial environments usually have continuous domain shifts i.e change in thematic structure, new vocabulary and different writing styles between the data trained on and the data that can be seen in production.

BERT[4] is a huge model with over 100 million parameters so not only would there be a need for a GPU to fine-tune the model but also at inference time multiple CPU's would not be enough which is not feasible for most businesses. According to [4] a BERT-large model for fine-tuning experiments would require over 16GB GPU memory proving extremely costly. Most success of BERT is based on transfer learning to a downstream task, however, the models that use BERT's pretraining are trained for general semantic comprehensions. Thus, for domain-specific fields such as medicine, law, finance and astronomy etc we can expect to see poor results for contextualized embeddings for domain-specific terms. One way to counter this issue would be to train BERT on a domain-specific corpus to truly encapsulate the contextuality of the embeddings such as done in BioBERT [12]. However, as discussed earlier creating a model with BERT for high quality results would require a huge corpus and a lot of GPU resources to train the model. For this reason, we prefer LSTM-based models with attention mechanism for production-based QA systems.

2. Related Work

The generic architecture for existing MRC based LSTM/RNN QA models is illustrated in Figure 2. The text passage and the question are represented as word vectors after being passed through the Embedding layer. These are then passed through to the Encoder layer which provides us with contextual semantic information. The Match layer then takes the contextualized semantic information from the Encoder layer and captures the interaction between the text passage and the question. This query-aware representation is then passed to the Answer layer which predicts the span

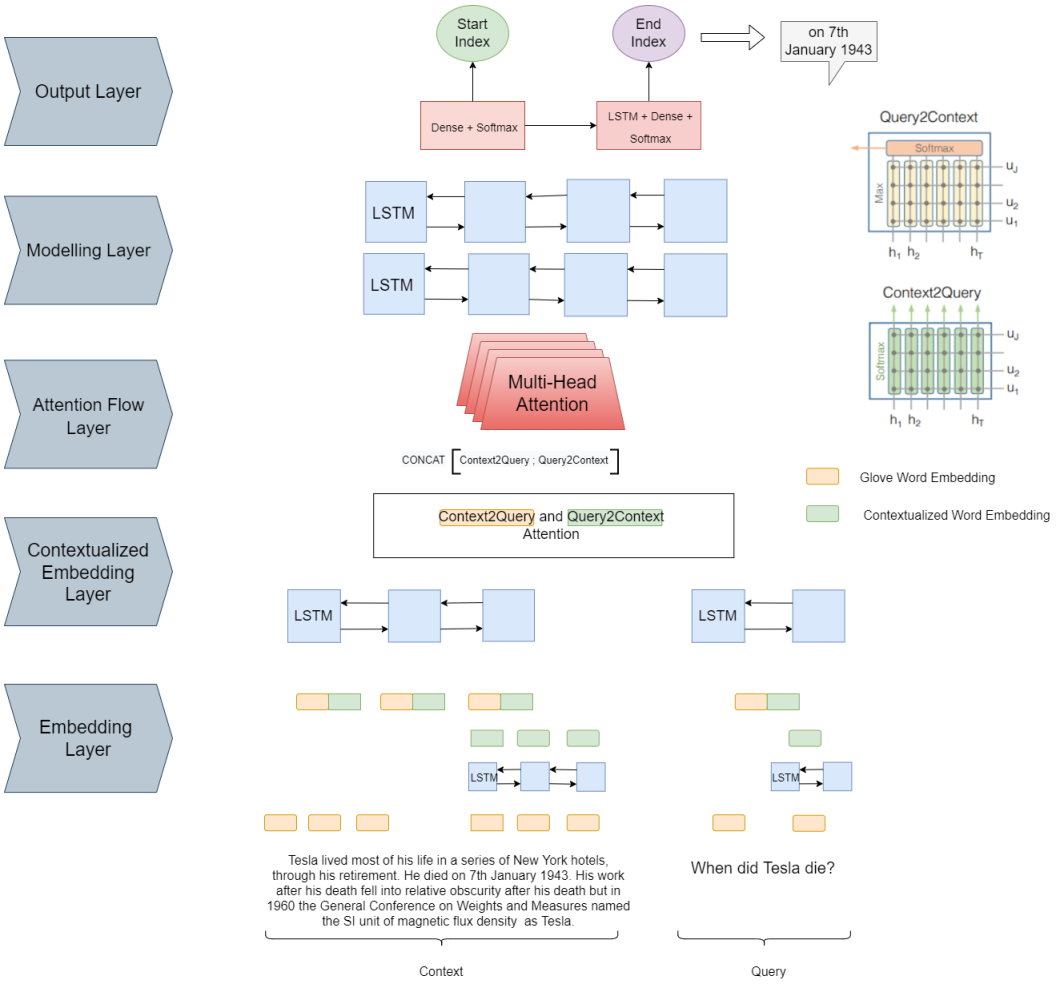

Figure 3. MuBAF Architecture.

of text.

The Match-LSTM Reader [13] makes use of the author's previous work in predicting textual entailment [14]. It attempts to solve the MRC problem by considering the query as the premise and the passage as the hypothesis. The Answer Pointer layer motivated by [15] then produces a sequence of answer tokens as the most suitable answer.

Another RNN-based model Reasoning Network (ReasoNet) [16] is designed to imitate the inference process of a normal human. Like a human it repeatedly reads the passage with attention on different times till it manages to find a satisfying answer. It take multiple turns to reason over the relationship and interaction between the passage, question and answer. Finally, it use reinforcement learning to dynamically determine whether to continue reading or to terminate.

Deep Residual Co-attention (DCN+) [17] model builds on the existing Dynamic Co-attention Network (DCN) [18]. They add a deep residual co-attention encoder to the network which allows the network to build richer input representations. Through stacking attention layers, the network also manages to model long range dependencies which prove extremely useful for longer questions. DCN+ also take inspiration from [19] which shows how skip layer connections facilitate signal propagation.

FusionNet's [20] claims that if all the information is utilized from the word embedding level to the highest semantic level representation it would would lead to more accurate answers. However, due to the surging complexity of using all these layers previous literature considers only partial information. To solve this limitation the paper introduces an attention scoring function which manages to utilize all layers representations without the issue of massive training. This fully-aware attention is a multi-level attention mechanism which understands the complete information in the question and exploits it layer by layer on the context side.

Lastly, we look at the BiDAF [8] model. The paper discusses how typically methods of uni-directional attention are used. BiDAF introduces attention mechanism in both directions, query-to-context and context-to-query, which

provide complementary information to each other. This bi-direction attention helps to obtain query-aware context representations.

3. Method

The MuBAF model builds on the existing BiDAF [8] model. Thus, there are some similar components which we have borrowed. We make significant modifications in all of the components within the architecture except the Contextualized Embedding Layer and the Modelling Layer. The architecture which is displayed in Figure 3 is a multi-stage process which consists of the following components:

Word Embedding Layer To represent the words in the context paragraph and query respectively, let $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_T\}$ and $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_T\}$. Firstly, we use a word embedding layer which makes use of two word embeddings. A word embedding maps each word to a high-dimensional vector space so as to be used as the input to our model. The first word embedding we use is the pre-trained word vector, GloVe [21] which has an embedding size of *d/2*, where *d* = 200. The second word embedding also of size *d/2* replaces the previously used CNN-based character-level embedding [22] with a contextualized word embedding which was inspired from the RNN/LSTM based word embedding ELMo [9]. The reason we went with a contextualized embedding over a character-based embedding was because it assigns each word a representation based on its context, thereby capturing uses of words across varied contexts. This proves extremely useful when we are trying to find the answer within the context of the passage. Although we didn't use ELMo, we sent individual word embeddings for each word of the query and context through a Bi-LSTM to get their contextualized representations. We then finally concatenate both these *d/2* dimensional embeddings to get a *d* dimensional embedding and send it to the next layer.

Contextualized Embedding Layer The concatenation of the contextualized word embedding and GloVe word embedding vectors is passed to a two-layer Highway Network [23]. This primarily acts as a gate flow which attempts to diminish the elements of the concatenated embedding which are less important and retain the elements which are more important. The input to the Highway Network is the concatenated word embedding for the context and query whose sizes are $d \times T$ and $d \times J$. The outputs of the Highway Network are two sequences of *d* dimensional vectors, or more conveniently, two matrices: $\boldsymbol{C} \in \mathbb{R}^{d \times T}$ for the context output and $\boldsymbol{Q} \in \mathbb{R}^{d \times J}$ for the query output. These matrices are then fed into a Bi-LSTM to model the temporal interactions between words. Since the LSTM is bidirectional we concatenate it's output to get to get two different outputs $\boldsymbol{H} \in \mathbb{R}^{2d \times T}$ and $\boldsymbol{U} \in \mathbb{R}^{2d \times J}$.

Multi-Head Attention Flow Layer This layer is responsible for linking and fusing information from the context and the query words. Unlike previously popular attention mechanisms [24, 25, 16], the attention flow layer is not used to summarize the query and context into single feature vectors. Instead, the attention vector at each time step, along with the embeddings from previous layers, are allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization.

The inputs to the Multi-Head Attention Flow Layer are our context $\boldsymbol{H}$ and query $\boldsymbol{U}$ matrices while the output is a query-aware vector representations $\boldsymbol{G}$. In this layer, we compute attentions in two directions: from context to query (C2Q) and query to context (Q2C). Both of these attentions are derived from a shared similarity matrix $\boldsymbol{S} \in \mathbb{R}^{T \times J}$ between the contextual embeddings layer outputs H and U, where $\boldsymbol{S}_{tj}$ indicates the similarity between t-th context word and j-th query word. The similarity matrix is computed by

$$\boldsymbol{S}_{tj} = \alpha(\boldsymbol{H}_{:t}, \boldsymbol{U}_{:j}) \in \mathbb{R}$$

where α is a trainable scalar function that encodes the similarity between t-th column vector of $\boldsymbol{H}$ and j-th column vector of $\boldsymbol{U}$. This similarity matrix $\boldsymbol{S}$ is now further used to obtain the attentions in both directions. Firstly, C2Q Attention is calculated which signifies which query words are most relevant to each context word. Secondly, Q2A is calculated to see which attention signifies which context words have the closest similarity to one of the query words and are hence critical for answering the query. Lastly, the

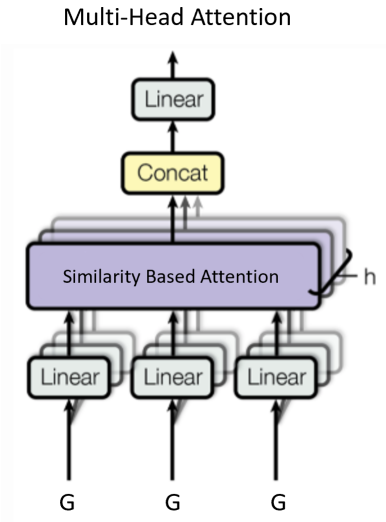

Figure 4. Multi-Head Attention for MuBAF.

four values are concatenated to give us our query-aware vector representations $\boldsymbol{G}$. These includes the contextual embedding $\boldsymbol{H}$, C2Q output, Hadamard product of $\boldsymbol{H}$ and

In 1870, Tesla moved to Karlovac, to attend school at the Higher Real Gymnasium, where he was profoundly influenced by a math teacher Martin Sekulic´. The classes were held in German, as it was a school within the Austro-Hungarian Military Frontier. Tesla was able to perform integral calculus in his head, which prompted his teachers to believe that he was cheating. He finished a four-year term in three years, graduating in 1873.	
1. In what language were the classes given?	German
2. Who was Tesla's main influence in Karlovac?	Martin Sekulic
3. Why did Tesla go to Karlovac?	attend school at the Higher Real Gymnasium

Table 1. An example taken from the SQuAD dataset showing a paragraph from Wikipedia which has 3 questions associated with it along with its corresponding answers

C2Q and finally the Hadamard product of $\boldsymbol{H}$ and Q2C. These together yield $\boldsymbol{G} \in \mathbb{R}^{8d}$ where each column vector of $\boldsymbol{G}$ can be considered as the query-aware representation of each context word. We then forward $\boldsymbol{G}$ through a multi-head attention layer where key, value and query inputs of the layer are all $\boldsymbol{G}$. This can be seen in Figure 4 which is a modification to the diagram of multi-head attention in [26] as it suits our problem and notation. Multi-head attention mechanism runs through an attention mechanism several times in parallel. These independent attention outputs are then concatenated and linearly transformed into the expected dimension $\mathbb{R}^{8d}$. Our inspiration to run the query-aware representation $\boldsymbol{G}$ through a multi-head attention layer is from [26] where multiple heads provide the model with various different aspects of the query-aware representation and improve its expressive ability allowing it to capture richer interpretations.

Modeling Layer The input to the modeling layer is the output of the multi-head attention layer $\boldsymbol{G} \in \mathbb{R}^{8d}$. The output of the modeling layer captures the interaction among the context words conditioned on the query. This is different from the contextual embedding layer which captures the interaction among context words independent of the query. In this layer we use a Bi-LSTM which has two layer which yields an output matrix $\boldsymbol{M} \in \mathbb{R}^{2d}$ which is passed onto the output layer to predict the answer.

Output Layer The MRC QA task requires the models to find a span within the paragraph to answer the query. The span is derived by predicting the start $\boldsymbol{p^1}$ and the end $\boldsymbol{p^2}$ indices of the phrase in the paragraph. To obtain the probability distribution of the start index over the entire paragraph we use the concatenation of our query-aware representation $\boldsymbol{G}$ which is the output of the multi-head attention layer and $\boldsymbol{M}$ which is the output of our modelling layer. We first send this concatenation matrix for the start index through a 5-layered Fully-Connected Network (FCN) and then take the softmax of the result to get the probability distribution over the entire passage. We then pass the output of the modelling layer $\boldsymbol{M}$ through another Bi-LSTM layer followed by a FCN and softmax to get the probability distribution of the ending index over the entire passage.

$$\boldsymbol{p^1} = \text{softmax}\Big(FCN(w_{(p^1)}[\boldsymbol{G};\boldsymbol{M}])\Big) \in \mathbb{R}^{10d}$$

$$\boldsymbol{p^2} = \text{softmax}\Big(FCN\left(BiLSTM(w_{(p^2)}[\boldsymbol{G};\boldsymbol{M}])\right)\Big) \in \mathbb{R}^{10d}$$

Finally, we set a condition that our ending index must come after our starting index to make sure the flow of span is correct. To calculate the loss of the entire model we take the sum of cross entropy losses for our starting and ending indices and use AdaDelta as our optimizer to achieve the results mentioned in 4

4. Experiments and Results Evaluation

For modern Question Answering (QA) models, there is a great need for well-constructed, high-quality and large-scale datasets. MRC can be divided into a total of 4 categories of datasets: Span-Prediction, Free-Form, Cloze-Style and Multi-Choice. We will primarily be working on Span-Prediction or otherwise known as extractive question answering. Here the machine looks within the passage and extracts a span of the text which is most suitable as an answer. Specifically, the model outputs the indices of the starting and ending position of the span as the answer to the posed question. The dataset that we will be working on is the following:

SQuAD: The all-famous Stanford Question Answering Dataset (SquAD) [6] consists of over 100k (Passage, Question and Answer) triplets which were collected from 536 Wikipedia passages. The training dataset consists of 87599 (passage, question and answer) triplets and the validation datset consists of 34726 triplets. A sample text passage along with the posed questions and answers is shown in Table 1. This specific dataset was the first unique one of its kind which has both variable length answers which were created by humans through crowd-sourcing. The answers are mostly located as a span within the text of the passage making this an extractive QA dataset.

For our experiments we plan to run our model on the dataset mentioned above and compare it with existing work on QA models mentioned in 2. Since we are trying to solve

Model	Batch Size	Optimizer	Learning Rate	Number of Heads	FC Layer	Contextual Embedding	EM (%)	F1 (%)
Base	16	AdaDelta	0.01	1	✗	✗	31.75	42.93
Run 1	16	AdaDelta	0.03	1	✗	✓	55.21	60.44
Run 2	16	Adam	0.03	8	✗	✓	42.25	46.63
Run 3	16	Adam	0.03	16	✗	✓	38.79	42.37
Run 4	16	Adam	0.03	4	✗	✓	43.81	47.34
Run 5	16	Adam	0.03	1	✓	✓	56.76	62.92
Run 6	8	Adam	0.03	1	✓	✓	54.54	59.32
Best	32	AdaDelta	0.03	1	✓	✓	**57.87**	**63.56**

Table 2. Results of exact Match (EM) and F1 scores of the multiple experiments ran while changing the hyper-parameters shown as the column names

the span-based prediction problem in MRC, we will look into SQuAD [6]. We will train and run our model on the SQuAD [6] dataset and test it separately on its validation set. We will then compare them with the existing models which make use of RNN/LSTM based architecture. Table 3 below shows the results for the models discussed in section 2 on the SQuAD dataset. The evaluation metrics we will use for both datasets are Exact Match (EM) and F1-Score (F1) for the QA tasks, where EM assigns a 1.0 if both the predicted answer and golden standard answer match otherwise 0.0. F1 then computes the harmonic mean (average) word overlap between both the predicted and golden standard answer where P and R are precision and recall respectively.

$$F1 = \frac{2 * P * R}{P + R}$$

For running our code we used the starter code [2] which is a PyTorch implementation of BiDAF [8]. We also use python's Flair library [3] for POS and NER Tags and Tokenization. After implementing changes in the architecture shown in Figure 3 and discussed in our contributions we ran the MuBAF model while changing the following hyper-parameters: Batch Size, Optimizer, Learning Rate, Number of Heads for the Multi-Head Attention Layer, presence of Fully Connected Network and Contextual Embedding. The results of our experiments can be visualized in Table 2. Let it be noted that these are only a few of the experiments that we ran containing different parameters and that many more experiments were run with multiple combinations of parameters.

The baseline of the starter code provides us with a baseline EM and F1 score of 31.75 and 42.93. Although BiDAF's original score are much higher, the original BiDAF code that was found online used the allennlp [4] python library. Since the library registers and loads the model directly we have no idea of the hyper-parameters used and replicating the results was not possible. So, after closely following the baseline code for the available starter code we confirmed that the math and the code were similar. Thus we ran the baseline code and our other experiments with the same number of epochs and tested out how our changed model architecture performed. Even though, the results didn't reach the expectations of the original results, we did see a huge improvement while working with some architectures over others. The use of the contextual embeddings and FCN at the end to predict the indices of the span proved to give higher results than the baseline. Our best run was using both the contextual embeddings along with the FCN without multi-head attention with a larger batch size. This gave us a EM and F1 score of 57.87 and 63.56 respectively. This was a huge improvement on the baseline we saw and is worth exploring these modified techniques with the original results of BiDAF.

Although, we were disappointed to see the overall model give low results we were pleased that the model did significantly improve on the baseline. We further observed that the multi-headed attention didn't work out as we expected it to. One reason for that may be because we were applying the multi-head attention to the concatenated output of C2Q and Q2C attention scores instead of sending them through the multi-head attention layer separately. The multi-head attention which attempts to find multiple representations could not find a rich representation with the C2Q and Q2C concatenated. Another reason for it not performing the way we had expected was because the multi-head attention was not provided with a mask or positional encoding which we see in the transformer architecture in [26].

[2] https://github.com/kushalj001/pytorch-question-answering/blob/master/2.%20BiDAF.ipynb

[3] https://github.com/flairNLP/flair

[4] https://github.com/allenai/allennlp

Reference	Model	Val Set		Test Set	
		EM	F1	EM	F1
Wang et al. [13]	Match-LSTM	67.6	76.8	67.9	77.0
Seo et al. [8]	BiDAF	67.7	77.3	68.0	77.3
Shen et al. [16]	ReasoNet	70.8	79.4	69.1	78.9
Xiong et al. [17]	DCN+	74.5	83.1	75.1	83.1
Huang et al. [20]	FusionNet	75.3	83.6	76.0	83.9

Table 3. Existing model performances on SQuAD[6]. Resuts sorted by Test F1 scores

5. Conclusion

In this paper we introduce the MuBAF model which builds upon the existing BiDAF QA model. Here we move back to using a RNN/LSTM based model due to the issues that BERT poses for QA at a production level. In this model we add and modify multiple components to the existing architecture to achieve more contextual, diverse and rich representations of our embeddings and our query-aware representation. Although, this model doesn't compete with the state-of-the-art QA systems we do see our model significantly improve on the baseline architecture of the BiDAF code we began with. We also came to the conclusion that the addition of the multi-head attention did not achieve the results we were hoping it would. Reasons we found were because the multi-head attention was applied to the concatenated query-aware representation instead of the separate attention scores. Furthermore, we didn't provide the multi-head attention with a masking matrix and positional encoding making it lose out on performance.

References

[1] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015. 1

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2

[5] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. 1, 2

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 1, 5, 6, 7

[7] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017. 1

[8] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016. 2, 3, 4, 6, 7

[9] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 2, 4

[10] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020. 2

[11] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. Luke: deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*, 2020. 2

[12] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020. 2

[13] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016. 3, 7

[14] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015. 3

[15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015. 3

[16] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1047–1055, 2017. 3, 4, 7

[17] Caiming Xiong, Victor Zhong, and Richard Socher. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*, 2017. 3, 7

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

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[20] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017. 3, 7

[21] 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. 4

[22] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. 4

[23] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. 4

[24] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. arxiv 2015. *arXiv preprint arXiv:1511.02301*, 2016. 4

[25] Alessandro Sordoni, Philip Bachman, Adam Trischler, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245*, 2016. 4

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 5, 6