Transformer with CSAN for Question Answering: SQuAD2.0

Simerjot Kaur
sk3391@stanford.edu
Default Project (Non-PCE)

Abstract

Current end-to-end machine reading and question answering models are based on sequential RNNs (primarily LSTMs) with attention mechanism. While these models have shown good progress on the machine reading and question answering tasks, these models are slow in both training and inference due to their sequential nature. In this project we use a novel approach to question and answering employed by the QANet authors by replacing sequential RNNs with convolutions and self-attentions as the building blocks of encoders. Further, in order to simultaneously capture model dependencies among local elements and neighboring spaces, QANet model was further extended to include 2D-Convolutional Self Attention (CSAN). Our trained QANet model was able to obtain an F1 score of 66.41 and EM score of 62.43 on Squad 2.0 dev dataset. Further, we obtained an F1 score of 64.07 and EM score of 60.02 for QANet with CSAN. Notably, the score obtained for QANet w/ CSAN is lower than QANet score because we could only use a small model and batch size when using QANet w/ CSAN due to the limited GPU memory available on NV6 machines.

1 Introduction

Current end-to-end machine reading and question answering models are based on sequential RNNs (primarily LSTMs) with attention mechanism. Most reading and question answering models employ two key ingredients: (a) recurrent models to process sequential inputs (b) an attention component to cope with long term interactions. While these models have shown good progress on the machine reading and question answering tasks, these models are slow in both training and inference due to their sequential nature. In this project we employed a novel approach to question and answering employed by QANet[9] authors: replacing sequential RNNs with convolutions and self-attentions as building blocks of encoders that separately encodes the query and context. The key motivation behind this design is that convolution captures the local structure of the text, while the self-attention learns the global interaction between each pair of words. In this project, we also enhance QANet by using convolutional self attention (CSAN) which uses the power of CNN on modeling localness of self attention (SAN) and offers the abilities to 1) capture neighboring dependencies, and 2) model the interactions between multiple attention heads.

Formally, we can describe the machine reading and question answering task as follows: Given a context paragraph with n words $C=\{c_1, c_2,\ldots, c_n\}$ and the query sentence with m words $Q=\{q_1, q_2, \ldots, q_m\}$, output a span $S=\{c_i, c_{i+1},\ldots, c_{i+j}\}$ from the original context paragraph C.

2 Related Work

This problem is an interesting task because it provides a measure for how well systems can 'understand' text. There have been a lot of research done in this area, first to model SQuAD 1.1 and now extended to SQuAD 2.0. The difference between the datasets is that SQuAD 2.0 contains no answer examples as well. The work in this area can be divided into two

distinct divisions: one for models that use pre-trained contextual embeddings (PCE), and another for non-PCE models only. PCE models like ELMo and BERT are based on the idea that to represent a piece of text, word embeddings that depend on the context in which the word appears in the text should be used. These models have been extremely successful for SQuAD and have achieved high performance. In the non-PCE world, new concept of transformers was recently introduced which replaces sequential RNNs with self-attention and convolution. This project is inspired by this novel approach and has tried to implement QANet (question answering using transformers) from scratch and has further enhanced it with CSAN to use the power of CNN on modeling localness of self attention also.

3 Approach

3.1 Baseline BiDAF

Since the default project has been implemented, the provided BiDAF[1] model has been used as baseline.

3.2 BiDAF with Character Embeddings

Since the baseline BiDAF was implemented based on word embeddings only, the first improvement was to incorporate character-level embeddings. In order to obtain fixed size vector representation of each word, convolutional network has been adopted and passed through a 2-layer highway network. The final embedding for BiDAF model, is concatenation of word and char embeddings.

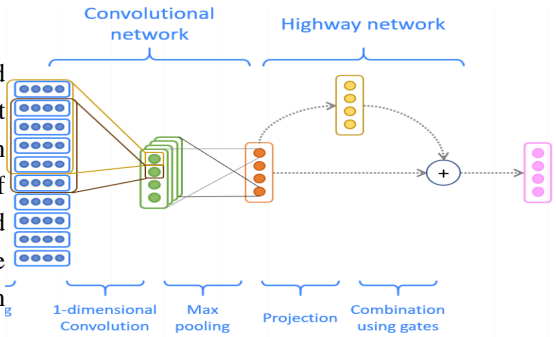


Figure 1: Character-based convolutional embedding

3.3 QANet

QANet architecture has next been implemented **from scratch**. The high level structure of the model consists of five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer:

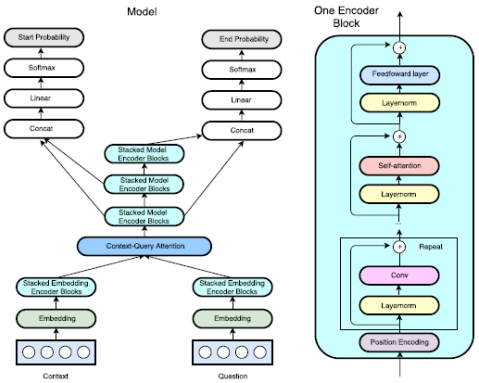


Figure 2: Overview of QANet architecture

1. **Input Embedding Layer:** This layer is similar to the BiDAF input embedding layer with output of the layer of dimension $p_1 + p_2$ (300+200 = 500).
2. **Embedding Encoder Layer:** This layer replaces the bidirectional LSTMs with stack of Position Encoding + 4 convolutional layers + self-attention layer + feed-forward layer.
    a. Positional Encoding: A positional encoding is added to the input at the beginning of each encoder layer consisting of sine and cosine functions at varying wavelengths.
    b. Convolution Layer: Model uses depthwise separable convolutions[4] with kernel size=7.
    c. Self-Attention Layer: This layer draws inspiration from Transformers[5]. It first creates 3 vectors Query, Key, and Value matrices from each embedding and then produce attention scores. Instead of performing single attention, transformer uses 8 attention heads.
    d. Feed-Forward Layer: This is a fully connected feed-forward network.

These layers are placed inside a residual block and layer-normalization[6] is performed.

3. **Context-Query Attention Layer:** In this layer similarities between each pair of context (C) and query (Q) words is computed and stored in matrix S[7]. For context-to-query attention (A),

each row of S is normalized producing $\bar{S}$ and hence $A = \bar{S} \bullet Q^T$. For query-to-context attention (B), each column of S is normalized producing $\bar{\bar{S}}$ and hence $B = \bar{S} \bullet \bar{\bar{S}}^T \bullet C^T$.

4. **Model Encoder Layer:** The input of this layer at each position is $[c, a, c \odot a, c \odot b]$, where a and b are respectively a row of attention matrix A and B. The layer parameters of model encoder layer is same as embedding encoder layer, but there are 2 convolutional layers within a block and within each model encoder layers there are 7 encoder blocks.

5. **Output Layer :**This layer predicts the probability of each position in the context being the start or end of an answer span.

$$p^1 = softmax\left(W_1[M_0; M_1]\right) \quad p^2 = softmax\left(W_2[M_0; M_2]\right)$$

where W_1 and W_2 are trainable variables and M_0, M_1, M_2 are the output of 3 model encoders.

The objective function is negative sum of log probabilities, obtained above, and averaged over all training examples:

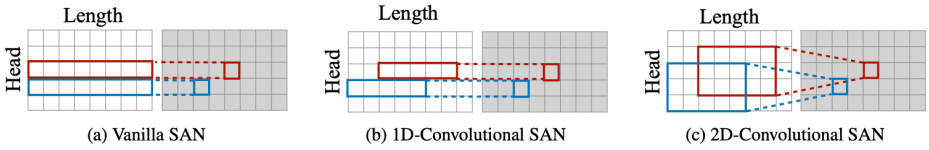
$$L(\theta) = -\frac{1}{N}\sum_{i}^{N}[\log(p^1_{y_i^1} + p^2_{y_i^2})]$$

where y_i^1 and y_i^2 are ground-truth starting and ending position of example i.

3.4 QANet with CSAN (Convolutional Self Attention)

Although Self Attention (SAN) has achieved significant improvements, it has two major limitations. Firstly, SAN fully takes into account all the signals with a weighted sum operation, which disperses the distribution of attention, which may result in overlooking the relation of neighboring signals. Secondly, the multi-headed attention perform attention heads independently, which misses the opportunity to exploit useful interactions across attention heads. To address these problems, we used a novel convolutional self-attention network (CSAN)[10] approach, leveraging power of CNN on modeling localness for SAN.

There are two proposed approaches to overcome the two limitations of SAN: 1) 1D-CSAN, as shown in Fig 3(b), in which window is assigned with width M where 1 < M < I, but height is consistently fixed to one; 2) 2D-CSAN, as shown in Fig 3(c), where window is assigned with unrestricted height N where 1 < N < H. QANet w/ CSAN has been implemented **from scratch**.



(a) Vanilla SAN (b) 1D-Convolutional SAN (c) 2D-Convolutional SAN

Figure 3: (a) Vanilla SAN, 1D- (b) and 2D- (c) Convolutional SAN

Given a window width M, the relevance between the i-th and j-th elements can be normalized according to the neighboring elements of i, instead of the whole input. Thus, the calculation of attention weight can updated as follows:

$$\alpha_{ij}^h = \frac{\exp e_{ij}^h}{\sum_{t=i-m}^{i+m} \exp e_{it}^h} \qquad e_{ij}^h = \lambda(x_i W_Q^h)(x_j W_K^h)^T \qquad m = \lfloor (M-1)/2 \rfloor$$

and 0 is padded when the index out of range. Accordingly, the output of attention operation can be revised as:

$$y_i^h = \sum_{j=i-m}^{i+m} \alpha_{ij}^h (x_j W_V^h)$$

Furthermore, 2D-CSAN model simultaneously model dependencies among local elements and neighboring subspaces. The 1-D attentive area (1 × M) can be expanded to a 2-D rectangle (N × M), which consists of both number of elements and number of heads. Consequently, the proposed model calculates the energy between the i-th element in the h-th head and the j-th element in the s-th head. Thus, the energy calculation can be updated as:

$$e_{ij}^{hs} = \lambda(x_i W_Q^h)(x_j W_K^s)^T$$

Accordingly, the energy normalization Equation (4) and the weighted sum of the elements Equation (5) can be respectively revised as:

$$\alpha_{ij}^{hs} = \frac{\exp e_{ij}^{hs}}{\sum_{k=h-n}^{h+n} \sum_{t=i-m}^{i+m} \exp e_{i,t}^{hk}}$$

$$y_i^h = \sum_{s=h-n}^{h+n} \sum_{j=i-m}^{i+m} \alpha_{ij}^{hs} (x_j W_V^s) \qquad n = \lfloor (N-1)/2 \rfloor$$

Thus, the attention distribution represents the dependencies among head and the output of each head covers different group of features. 2D-CSAN equals to 1D-CSAN when N = 1.

4 Experiments

4.1 Data

The dataset for the project is SQuAD 2.0 which has been provided as part of the default class project setup. The SQuAD 2.0 dataset has three splits: train, dev and test.

- Train(129,941 examples):All taken from the official SQuAD 2.0 training set.
- Dev(6078 examples):Roughly half of the official dev set, randomly selected.
- Test(5915 examples):Remaining examples of official dev set, plus hand-labeled examples.

4.2 Evaluation Method

To evaluate the model Exact Match (EM) and F1 scores have been used as metric. **Exact Match** is binary measure of whether system output matches ground truth answer exactly. **F1** is the harmonic mean of precision and recall = 2×prediction×recall/(precision+recall).

4.3 Experimental Details

The table below summarizes the various key training details for all the four models used:

	BiDAF	BiDAF w/ Char Emb	QANet	QANet w/ CSAN
Optimizer Used	AdaDelta	AdaDelta	Adam with β_1=0.8, β_2= 0.999	Adam with β_1=0.8, β_2= 0.999
Batch Size	64	64	32	8
Number of Epochs	30	30	30	30
Learning Rate	0.5	0.5	Inverse exponential increase from 0.0 to 0.001 in first 1000 steps, then constant LR for remainder of training	Inverse exponential increase from 0.0 to 0.001 in first 1000 steps, then constant LR for remainder of training
Hidden Size	100	100	96	64
EMA Decay	0.999	0.999	0.9999	0.9999
Dropout Probability	0.2	0.2	Dropout rates on word, character embeddings are 0.1 and 0.05, the dropout rate b/ w two layers is 0.1. Adopted stochastic depth method within each encoder layer, sublayer ℓ has survival prob p_ℓ = 1− ℓ / L (1−p_L) where L is the last layer and p_L = 0.9.	Dropout rates on word, character embeddings are 0.1 and 0.05, the dropout rate b/ w two layers is 0.1. Adopted stochastic depth method within each encoder layer, sublayer ℓ has survival prob p_ℓ = 1− ℓ / L (1−p_L) where L is the last layer and p_L = 0.9.
L2 Weight Decay	0	0	3*10-7	3*10-7
Loss Function	NLL	NLL	NLL	NLL

4.4 Results (Non-PCE)

Since, we are not using the pre-trained BERT Models, we have submitted our results to the **Non-PCE leaderboard**. On the **Test Non-PCE Leaderboard** we achieved **EM** score of **60.507** and **F1** score of **64.099.** On the **Dev Non-PCE Leadership** we achieved **EM** score of **62.393** and **F1** score of **66.405**.

The table below compares results achieved on dev set using baseline BiDAF model, BiDAF model augmented with character embedding, QANet architecture and QANet w/ CSAN.

	BiDAF	BiDAF w/ Char Emb	QANet	QANet w/ CSAN
F1-Score	59.89	62.59	66.41	64.07
EM-Score	56.01	59.28	62.43	60.02

4.4.1 Baseline BiDAF vs BiDAF with Character Embeddings

The plot below compares the performance achieved on the dev set using BiDAF with character embedding (*Red*) and baseline BiDAF model (*Blue*). As expected, BiDAF with character embedding outperforms the BiDAF model. The character embeddings are able to capture out of vocabulary words, hence improving the model's performance.

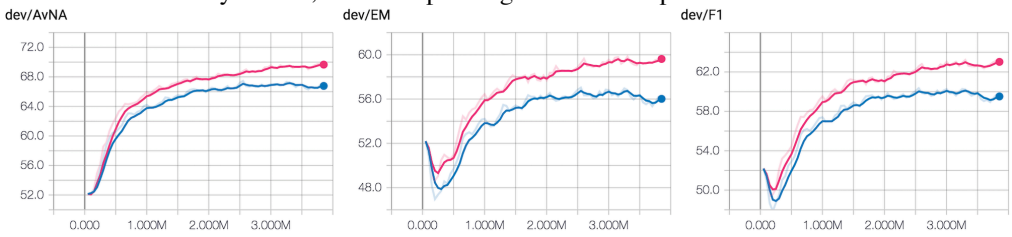


Figure 4: Dev Set performance comparison between BiDAF (Blue curve) and BiDAF with Char Embedding (Red Curve)

4.4.2 QANet

The plot below shows the dev set performance achieved using QANet architecture. As can be observed that QANet significantly outperforms BiDAF as well as BiDAF w/ Character Embeddings. This improvement was expected because QANet not only learns the relationships between words and order of words in the passage and corresponding questions, but also where the answer appears in the passage with respect to those relationships.

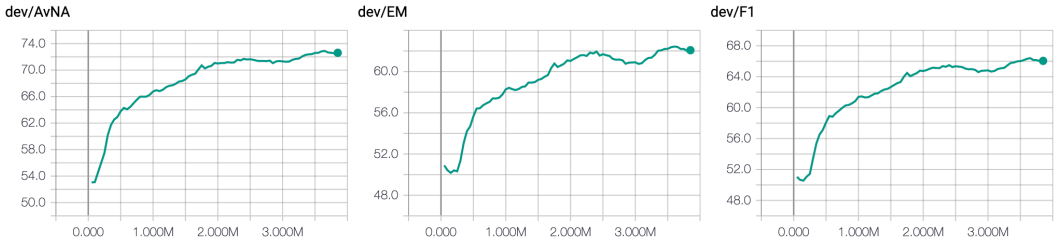


Figure 5: Dev Set performance of QANet

4.4.3 QANet with CSAN

The plot below shows the dev set performance achieved using QANet with CSAN. Since CSAN tries to capture simultaneously the model dependencies among local contexts as well as neighboring subspaces (dependencies among the attention heads), we expected CSAN will outperform the vanilla QANet. However, the below graphs depict otherwise. This is because in order to capture the dependencies in CSAN there is an increase in the amount of GPU memory needed, which limits the maximum batch size we could implement on NV12 machine to 8 with a hidden size of model as 64. This significantly reduces the achievable performance, but can easily be improved further by using higher memory GPUs for

computation, and partly re-optimizing the code for memory usage.

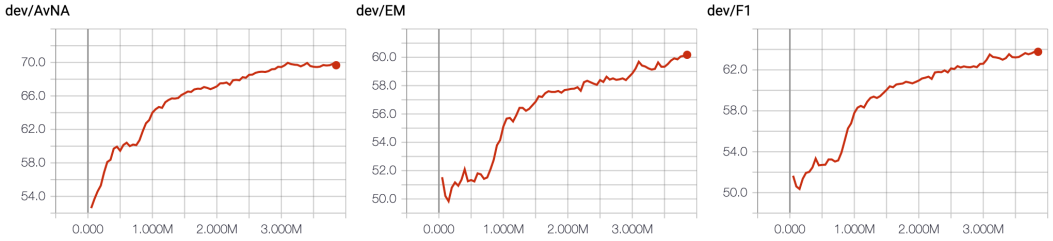


Figure 6: Dev Set performance of QANet with CSAN

4.4.4 Effects of Hyper parameters on Performance

Different hyper parameters like batch size, learning rate, loss function, stochastic dropout, optimization techniques and hidden size were analyzed to obtain the model's best performance. It was observed that the model performed best with batch size = 32, Learning Rate = 0.001, Loss function = Negative Log-Likelihood, Stochastic Dropout = Yes, Adam optimization and hidden size = 96.

Batch Size	32	16
F1	66.41	64.76
EM	62.43	60.81

Learning Rate	0.0005	0.001
F1	64.39	66.41
EM	60.53	62.43

Loss Function	NLL	Cross Entropy
F1	66.41	65.30
EM	62.43	61.97

Stochastic Dropout	No	Yes
F1	63.59	66.41
EM	60.52	62.43

Optimizer	AdaDelta	Adam
F1	62.38	66.41
EM	59.32	62.43

Hidden Size	64	96
F1	64.63	66.41
EM	60.78	62.43

5 Analysis

In order to perform qualitative evaluation, the analysis of inspecting key characteristics of the QANet model have been broken down into four subsections:

5.1 Error Analysis on Selected Examples

After training the model, 100 examples were selected to analyze the results produced by the model vs actual answers. There were 53 examples for which the actual answers were N/A (no answer) and 47 examples had the answer in the context paragraph. There were two key observations:

- Actual Answer = N/A: The model produced 68% correct answers i.e. N/A. For the rest 32%, the model produced wrong written answers.
- Actual Answer = Some Text: The model produced 70% correct answers and 30% wrong answers (N/A or wrong answer). Also, in case of the wrong answers, the model produces N/A 40% of the time and wrong answer 60% of the time.

After observing the questions, it was observed that the maximum wrong answers occur in the cases when questions start with 'what' (41%) and 'how' (35%). Fewer wrong answers occurred in the cases when questions start with 'when' (6%) and 'who' (18%). The most common types of errors made by the model can be broken down into four categories:

5.1.1 Long Term Dependency

The most common error observed was that the model was not able to perform long term dependency. For instance, In the example below, question was regarding Sybilla as to what did she introduce to Scotland. Since she was the wife of King David I's brother, the model was not able to capture this relationship and hence the question that the model answered was as to what King David introduced to Scotland rather than Sybilla.

step 1,650,396

- **Question:** What did Sybilla of Normandy introduce to Scotland?
- **Context:** Normans came into Scotland, building castles and founding noble families who would provide some future kings, such as Robert the Bruce, as well as founding a considerable number of the Scottish clans. King David I of Scotland, whose elder brother Alexander I had married Sybilla of Normandy, was instrumental in introducing Normans and Norman culture to Scotland, part of the process some scholars call the "Davidian Revolution". Having spent time at the court of Henry I of England (married to David's sister Maud of Scotland), and needing them to wrestle the kingdom from his half-brother Máel Coluim mac Alaxandair, David had to reward many with lands. The process was continued under David's successors, most intensely of all under William the Lion. The Norman-derived feudal system was applied in varying degrees to most of Scotland. Scottish families of the names Bruce, Gray, Ramsay, Fraser, Ogilvie, Montgomery, Sinclair, Pollock, Burnard, Douglas and Gordon to name but a few, and including the later royal House of Stewart, can all be traced back to Norman ancestry.
- **Answer:** N/A
- **Prediction:** Normans and Norman culture

The model can be improved by introducing self attention of phrases (n-gram words) rather than on single words.

5.1.2 Out of Vocabulary Words

Another error that was observed, was regarding out of vocabulary (OOV) words. The word “non-Muslims” in the question is most likely OOV word which it was not able to interpret. Although character embedding has been introduced, it was not able to understand the “non-Muslims” word, hence while concatenating the characters it dropped ‘non’ and interpreted as to how many Muslims are in Greater London, for which the model gave the answer of.

step 3,850,913

- **Question:** How many non-Muslims are in Greater London?
- **Context:** Greater London has over 900,000 Muslims, (most of South Asian origins and concentrated in the East London boroughs of Newham, Tower Hamlets and Waltham Forest), and among them are some with a strong Islamist outlook. Their presence, combined with a perceived British policy of allowing them free rein, heightened by exposés such as the 2007 Channel 4 documentary programme Undercover Mosque, has given rise to the term Londonistan. Following the 9/11 attacks, however, Abu Hamza al-Masri, the imam of the Finsbury Park Mosque, was arrested and charged with incitement to terrorism which has caused many Islamists to leave the UK to avoid internment.[citation needed]
- **Answer:** N/A
- **Prediction:** 900,000

The model can be improved by using byte level embedding or pertained BERT embeddings.

5.1.3 Lexical Gap

Another challenge in natural language is that same meaning can be expressed in different ways. Because question can usually only be answered if every referred concept is identified, in below example too, model failed to produce correct answer. The question was what can exhaust steam not fully do if the exhaust event is ‘insufficiently long’. In context paragraph, answer is given in terms of ‘if the exhaust even is too brief’. The model was not able to relate ‘insufficiently long’ and ‘too brief’, hence resulting in the wrong answer.

step 200,053

- **Question:** What can the exhaust steam not fully do when the exhaust event is insufficiently long?
- **Context:** The simplest valve gears give events of fixed length during the engine cycle and often make the engine rotate in only one direction. Most however have a reversing mechanism which additionally can provide means for saving steam as speed and momentum are gained by gradually "shortening the cutoff" or rather, shortening the admission event; this in turn proportionately lengthens the expansion period. However, as one and the same valve usually controls both steam flows, a short cutoff at admission adversely affects the exhaust and compression periods which should ideally always be kept fairly constant; if the exhaust event is too brief, the totality of the exhaust steam cannot evacuate the cylinder, choking it and giving excessive compression ("kick back").[citation needed]
- **Answer:** evacuate the cylinder
- **Prediction:** N/A

The model can be improved by providing synonym phrases in vocabulary or training the model on how a synonym is same as using ‘not’ of antonym.

5.1.4 Multilingualism

Finally, since there is not a single language that is always used, the model is expected to recognize a language and get the results on the go. In this example, although the answer looks to be correct, but it was depicted in Chinese language since the context contains ‘Chinese:’. Since the model doesn’t understand what pinyin is, it’s not able to predict the English version of the answer.

step 2,500,591

- **Question:** What is the Chinese name for the Yuan dynasty?
- **Context:** The Yuan dynasty (Chinese: 元朝; pinyin: Yuán Cháo), officially the Great Yuan (Chinese: 大元; pinyin: Dà Yuán; Mongolian: Yehe Yuan Ulus[a]), was the empire or ruling dynasty of China established by Kublai Khan, leader of the Mongolian Borjigin clan. Although the Mongols had ruled territories including today's North China for decades, it was not until 1271 that Kublai Khan officially proclaimed the dynasty in the traditional Chinese style. His realm was, by this point, isolated from the other khanates and controlled most of present-day China and its surrounding areas, including modern Mongolia and Korea. It was the first foreign dynasty to rule all of China and lasted until 1368, after which its Genghisid rulers returned to their Mongolian homeland and continued to rule the Northern Yuan dynasty. Some of the Mongolian Emperors of the Yuan mastered the Chinese language, while others only used their native language (i.e. Mongolian) and the 'Phags-pa script.
- **Answer:** Yuán Cháo
- **Prediction:** 元朝

The model can be improved by training the model on multilingual dictionary, so that the model can understand the meanings of multilingual words.

5.2 Ablation Study

The ablation study is performed on the two main components of the model: the number of heads used in Self Attention and number of encoder layers used in the Model Encoder Layer. As can be seen from the table, the number of encoder layers used in Model Encoder Layer is crucial. Changing the number of encoder layers from 4 to 8 contributes towards a gain of 2.1/1.9 in F1 and EM scores. This makes the number of encoder layers crucial as they are trying to capture local structure as well as global interactions between text. On the other hand, while the number of independent heads do contribute to gains in F1 and EM score but using lower number of heads doesn't hamper the model too much. This is because the heads are concatenated independently, hence there is no interdependency which might effect the model a lot.

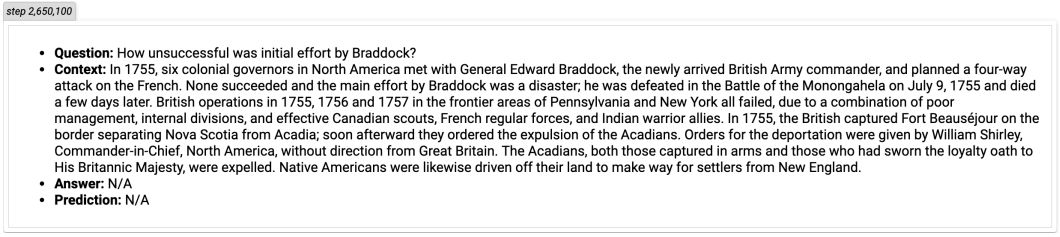
# Encoder Layers	3	7
F1	64.31	66.41
EM	60.53	62.43

# of Heads*	4	8
F1	64.76	63.6
EM	60.81	59.7

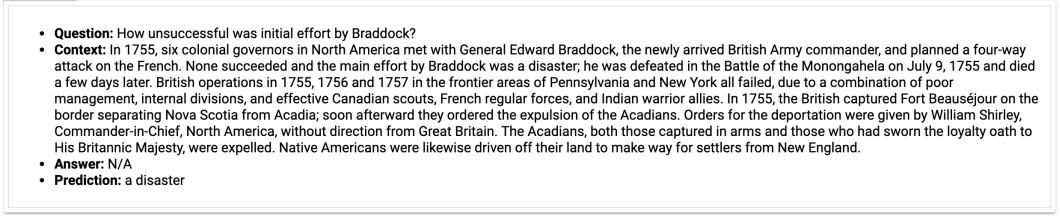
* In order to test number of heads = 8, the model had to be run on batch size of 16 to deal with CUDA out of memory error.

5.3 QANet vs QANet-CSAN

The models QANet and QANet-CSAN are also compared based on the answers provided. As can be observed QANet-CSAN is able to better capture lexical gaps than QANet. For instance, in the context below, there is nothing stated regarding how unsuccessful was Braddock initial effort. QANet-CSAN was able to interpret 'None succeeded and the main effort by Braddock was a disaster' and hence gave the correct result as N/A.



But as can be observed below, QANet was not so successful in interpreting the full phrase and hence resulted in wrong answer. This shows that QANet-CSAN can easily outperform QANet if the current implementation is improved further by using higher memory GPUs for computation, and partly re-optimizing the code for memory usage.



6 Conclusion and Future Work

In this project we used a novel approach to question and answering employed by the QANet authors by replacing sequential RNNs with convolutions and self-attentions as the building blocks of encoders. Further, in order to simultaneously capture model dependencies among local elements and neighboring spaces, QANet model was further extended to include 2D-Convolutional Self Attention (CSAN). Our trained QANet model was able to obtain an F1 score of 66.41 and EM score of 62.43 on Squad 2.0 dev dataset. Further, we obtained an F1 score of 64.07 and EM score of 60.02 for QANet with CSAN. The score obtained for

QANet w/ CSAN is lower than QANet score because we could only use a small model and batch size when using QANet w/CSAN due to the limited GPU memory available on NV6 machines. Hence this can be further improved by optimizing the code for memory usage and also using higher memory GPUs for computation. The model can be further improved by introducing self attention of phrases (n-gram words) rather than on single words. Also using byte level embedding or pertained BERT embeddings can further help in dealing with Out of Vocabulary words.

References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Ferhadi, Hananneh Hajishirzi. BiDAF (Bi-Directional Attention Flow for Machine Comprehension), ICLR 2017: https://arxiv.org/pdf/1611.01603.pdf

[2] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial, 2017: https://github.com/tensorflow/nmt

[3] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016: https://arxiv.org/pdf/1609.08144.pdf

[4] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 2017: https://arxiv.org/pdf/1610.02357.pdf

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, NIPS 2017: https://arxiv.org/pdf/1706.03762.pdf

[6] Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016: https://arxiv.org/pdf/1607.06450.pdf

[7] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering, ICLR 2017: https://arxiv.org/pdf/1611.01604.pdf

[8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context, 2019: https://arxiv.org/pdf/1901.02860.pdf

[9] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, Quoc V. Le. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension, ICLR 2018: https://arxiv.org/pdf/1804.09541.pdf

[10] Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, Zhaopeng Tu. Convolutional Self-Attention Network, cs.CL 2018: https://arxiv.org/pdf/1810.13320.pdf