Re-attention vs. Bi-attention: Investigating two Alignment Architectures for Neural Question Answering

Stanford CS224N Default IID Project

Ghaith Arar
Stanford University
ghaith01@stanford.edu

Abstract

In this project, I investigated how re-attention and bi-attention mechanisms focus on relevant information in Neural Machine Comprehension Question Answering models. I implemented a Mnemonic Reader which uses re-attention alignment architecture and compared multiple aspects with a BiDAF baseline network that uses bi-attention alignment. I incrementally added features to context and question embeddings to investigate their effects on the performance of the model.

1 Key Information

- Mentor: Lucia Zheng
- External Collaborators: N/A
- Sharing project: N/A

2 Introduction

Machine Comprehension Question Answering systems aim to answer questions given a context passage. Various attention mechanisms have been proposed and used successfully in such systems. Some of the state-of-the-art models use attention mechanism exclusively, while the common pattern in many other models is to use attention in combination with some type of recurrent neural network (RNN). However, Attention remains the backbone of almost all question answering architectures. In the context of Reading Comprehension, attention seeks to attend the question into the context paragraph to find the answer if exists.

The goal of this project is to investigate how Re-attention in Mnemonic Reader architecture differ from Bi-attention in Bi-Directional Attention Flow (BiDAF) network and how each seek to find relevant knowledge, and to gain a deeper understanding of their implementation and effectiveness.

I implemented Mnemonic Reader [1] and compared its performance against the provided baseline. I implemented a character-level embedding and added it to the BiDAF baseline. The BiDAF baseline model was trained and the results were used for comparison. I performed multiple experiments on the Mnemonic Reader model to explore how different aspects affect the performance of the model. A normalized term frequency (TF) and a binary exact context-question match were added incrementally to evaluate how they change the behaviour of the model.

3 Related Work

Seo *et al.* [2] proposed Bi-Directional Attention Flow (BiDAF) network, a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity. It uses bi-directional attention flow to obtain a query-aware context representation.

Xiong $et\ al.$[3] proposed a co-attention which is two-way attention between the context and the question. However, unlike Bidirectional Attention Flow, co-attention involves a second-level attention computation – i.e., attending over representations that are themselves attention outputs. Both approaches are single-round alignment architecture.

Huang $et\ al.$[4] and Xiong et al.[3] proposed multi-round alignment architectures that compute attentions repeatedly but in these approaches, the one attention is not aware of which parts of the context and question have been focused on in earlier attentions.

Hu $et\ al.$ [1] proposed a re-attention mechanism that temporally memorizes past attentions and uses them to refine current attentions in a multi-round alignment architecture. The idea is that words tend to share similar semantics when their attentions about same texts are highly overlapped. This leads to attention become either concentrated if previous attentions attend to same words of input, or distracted to attend to other words when previous attentions are not overlapped.

Chen $et\ al.$[5] showed that adding additional features to the input embeddings turned out to be extremely helpful. One of the features is exact match which is a simple binary feature indicates whether context words can be matched to one question word in question. Token features like (normalized) term frequency (TF) showed to improve question answering performance.

4 Approach

4.1 The Bi-Directional Attention Flow (BiDAF) Baseline.

I used the BiDAF model provided by CS-224N staff as the baseline for this project. The model uses 300-d GloVE word embeddings. I implemented an RNN-based GRU charactor embedding layer. The model was trained on the provided modified SQuAD 2.0 dataset and the result was used for comparison.

Original BiDAF Character Embedding Layer. The (BiDAF) architecture utilizes a Convolutional Neural Networks (CNN) to obtain character embedding. Charactor embedding vectors are treated as 1D input to the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. I decided to try a different approach for character embedding layer.

RNN-based Character Embedding Layer. Inspired by R-Net[6], I implemented the character embedding layer of Gated Self-Matching Networks[6] which consists of 1-layer recurrent neural network (RNN). Embeddings are obtained by by taking the final hidden states of a bi-directional (RNN). I implemented the layer using Gated recurrent units (GRU)[7] with $dim = 100$ for the hidden state since it is theoretically less computationally intensive than Long short-term memory (LSTM).

Single-round Bi-attention Mechanism. Bi-directional attention was proposed in BiDAF architecture paper[2]. Bi-attention allow the attention vectors at each time step to move through to the subsequent layer. This approach maintains more information and to some extend prevent early summarizing. However, this type of single-round attention is agnostic to attention in previous steps.

4.2 Mnemonic Reader

Multi-round Attention Architectures. Multi-round Alignment Architectures consist of multiple similar attention layers stacked on top of each other. Let $V^t = \{v_i^t\}_{i=1}^n$ and a passage $U^t = \{u_j^t\}_{j=1}^m$ denote the hidden representations of question and context in t-th layer, and $H^t = \{h_j^t\}_{j=1}^m$ is the corresponding question-aware context representation. Then the two similarity matrices can be computed as

$$E_i{}_j^t = f(v_i^t, u_j^t)\ , B_i{}_j^t = 1_{i \neq j} f(h_i^t, h_j^t)$$

However, this type of attention is not aware of previous attention in such architecture. The limitation of this type comes from the fact that attentive information can only flow to the subsequent layer through the hidden representation.

Re-attention Mechanism. Re-attention was proposed in Reinforced Mnemonic Reader paper [1]. This type of attention aims to remember previous attention and utilize them to fine-tune current attention. The motivation is that two words should be correlated if their attentions about same texts are highly overlapped, and be less related vice versa. The computation of re-attention is defined as follows. Let E^{t-1} and B^{t-1} denote the past similarity matrices that are temporally memorized. The refined similarity matrix $E^t (t > 1)$ is computed as

$$\hat{\mathrm{E}}_{ij}^t = Softmax(E_{i:}^{t-1}).Softmax(B_{j:}^{t-1})$$
$$\mathrm{E}_{ij}^t = f(v_i^t, u_j^t) + \gamma \hat{E}_{ij}^t$$

Where γ is a trainable parameter. And $Softmax(E_{i:}^{t-1})$ is the past context attention distribution for the $i-$th question word and $Softmax(B_{j:}^{t-1})$ is the self attention distribution for the $j-$th context word. When there is no overlap between two distributions, the dot product will be 0. On the other hand, if the two distributions are identical and focus on one single word, it will have a maximum value of 1. Hence, the similarity of two words can be explicitly measured using their past attentions. Since the dot product is relatively small than the original similarity, the γ is initialized with a tunable hyper-parameter and keet trainable. The refined similarity matrix can then be normalized for attending the question. Similarly, we can compute the refined matrix B^t to get the unnormalized self reattention as

$$\hat{\mathrm{B}}_{ij}^t = Softmax(B_{i:}^{t-1}).Softmax(B_{j:}^{t-1})$$
$$\mathrm{B}_{ij}^t = 1_{i \neq j}(f(h_i^t, h_j^t) + \gamma \hat{B}_{ij}^t)$$

End-to-end Mnemonic Reader Architecture. The Mnemonic Reader Architecture consists of three components as shown in Figure 1.

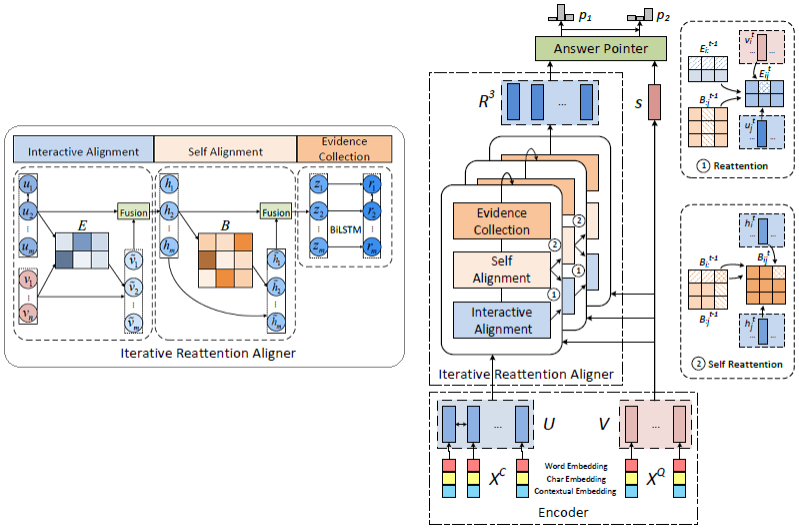

Figure 1: Overview of Mnemonic Reader Architecture. Different colors in E and B represent different degrees of similarity

1. **An encoder**: The encoder firstly converts each word to an input vector. The architecture uses the $100-dim$ GloVe embedding and $1024-dim$ ELMo[8] embedding. However, since CS224N project rules don't permit using any pre-trained weights other than GLoVe, I used $300-dim$ GloVe[9] for word-level embedding.
   The character-level embedding is obtained by encoding the character sequence with a bi-directional long short-term memory network (BiLSTM), where two last hidden states are concatenated to form the embedding. I kept both word-level and character-level embedding fixed during training.
   Additionally, I incrementally implemented a binary feature of exact match between words in context and question. Unlike the binary feature suggested by Chen $et\ al.$[5], my

implementation has only 1 dim that indicates a context word is in the question when it's 1 and 0 otherwise. A similar binary exact match was added to the question as well. I hypothesized that this binary feature will assist attention mechanism during training.
I implemented a normalized term frequency feature which was trained on the entire training data set before the main model was trained. Although [5] showed the effectiveness of deploying Part of Speech (POS) and Names Entity Recognition (NER) for both question and context, I did not implement these components since they would require using pre-trained weights.

Both $X^Q = \{\mathrm{x}_\mathrm{i}^\mathrm{q}\}_i^n$ and $X^C = \left\{\mathrm{x}_\mathrm{j}^\mathrm{c}\right\}_j^m$ are obtained through modeling each word with it's contextual information through a weight-sharing BiLSTM.

$$v_i = BiLSTM(x_i^q)\ ,\ u_j = BiLSTM(x_j^c)$$

denoted as two matrices $V = [v_1, ..., v_n] \in \mathbb{R}^{2d \times n}$ and $U = [u_1, ..., u_m] \in \mathbb{R}^{2d \times m}$

2. **An iterative aligner with the re-attention mechanism:** The iterative aligner consists of a stack of three aligning blocks. Each block consists of three modules as shown in Figure 1.

   - An interactive alignment.
   - A self alignment to attend the context against itself.
   - An evidence collection to model the context representation with a BiLSTM.

   The iterative aligner module aims to attend the question into the context. First, the similarity matrix $E \in R^{n \times m}$ is computed where the multiplicative product with non-linearity is applied as attention function. $f(u, v) = relu(W_u u)^\top relu(W_v v)$. The question attention for the j-th context word is then: $Softmax(E_{:j})$ which is used to compute an attended question vector $\hat{v_j} = V.Softmax(E_{:j})$. And to efficiently fuse the attentive information into the context, an heuristic fusion function, denoted as $o = fusion(x; y)$, is proposed as

$$\hat{x} = relu(W_r[x; y; x \circ y; x - y])$$
$$g = \sigma(W_g[x; y; x \circ y; x - y])$$
$$o = g \circ \hat{x} + (1 - g) \circ x$$

   The architecture stacks multiple aligning blocks with the re-attention mechanism to improve the ability of capturing complex interactions among inputs.

$$R^1, Z^1, E^1, B^1 = align^1(U, V)$$
$$R^2, Z^2, E^2, B^2 = align^2(R^1, V, E^1, B^1)$$
$$R^3, Z^3, E^3, B^3 = align^3(R^2, V, E^2, B^2, Z^1, Z^2)$$

   where align^t denote the t-th block. In the t-th block $(t > 1)$ and the hidden representation of question is fixed.

3. **An answer pointer:** The question representation V is summarized into a fixed-size summary vector s. Then the start probability $P_1(i)$ is computed by heuristically attending the context representation R^3 with the question summary.

$$P_1(i) = \propto exp(W_1^\top tanh(W_1[r_i^3; s; r_i^3 \circ s; r_i^3 - s]))$$

   Next, a new question summary $\hat{s}$ is updated by fusing context information of the start position.

$$P_2(j|i) = \propto exp(W_2^\top tanh(W_2[r_j^3; \hat{s}; r_j^3 \circ \hat{s}; r_j^3 - \hat{s}]))$$

   Full implementation details in the original paper [1].

5 Experiments

5.1 Data

I used the custom version of the Stanford Question Answering Dataset SQuAD 2.0 provided by CS224N staff, which contains both answerable and unanswerable questions, to train and test the

models. The training set has nearly 130K examples in the official SQUAD 2.0 training set. The development set and test set are roughly equally sized halves of the official SQUAD 2.0 development set, with about 6K examples each, since the official test set is hidden from the public.

5.2 Evaluation method

I evaluated on the F1 and EM (Exact Match) scores. The AVNA (Answer vs No Answer) was also included to better understand the model's classification accuracy considering only the answer (any span predicted) vs no-answer predictions.

5.3 Experimental details

Learning Rate (lr)**.** I experimented with fixed $lr \in \{0.5, 0.6, 1\}$. I also utilized a multi-step scheduler to decay the learning rate of all parameter groups by $\gamma = 0.9$ every two epochs.

Optimization I used Adadelta algorithm for the BiDAF model. For Mnemonic Reader model I experimented with Adadelta and Adam optimizers.

Hidden Size Throughout the experiment, I varied the hidden size of the architecture. Mostly the model perform best with hidden size between 64 and 128. I didn't observe any noticeable performance gain with increasing the hidden size beyond 150.

Batch Size Different batch sizes between 30 and 70 were tested within the GPU memory restraint. In general, I observed better performance with batch size equal or less than 45.

Training Device I performed all experiments locally on Nvidia GPU with Ampere architecture and 6 GB of GPU RAM.

5.4 Results

EM and F1 The table below shows the results of best performing experiments on both the BiDAF baseline with GRU-based character embeddings and my implementation of Mnemonic Reader.

Model	Variant	Dev EM	Dev F1	AvNA	Test EM	Test F1
BiDAF	B=64, dp=0.2, hs=100	59.84	62.85	69.61	–	–
BiDAF	B=64, dp=0.25, hs=120	59.25	62.78	68.95	–	–
Mnemonic Reader	B=45, dp=0.2, hs=100	59.88	62.96	68.44	–	–
Mnemonic Reader	B=45, dp=0.2, hs=100, TF, EX	60.36	63.16	68.93	**59.37**	**62.77**

My implementation of Mnemonic Reader gained roughly one point on both EM and F1 scores compared with the BiDAF baseline with GRU-based character embedding.

6 Analysis

Performance The Mnemonic Reader paper[1] reported EM score of 78.9 and F1 score of 86.3 on Dev set of SQuAD dataset. My implementation of Mnemonic Reader was trained and evaluated against a modified SQuAD 2.0 dataset. This modified dataset differs than SQuAD dataset in that it contains unanswerable questions.

The Mnemonic Reader paper uses 100-dim GloVe embedding and 1024-ELMo pretrained embedding in addition to POS and NER embedding. My implementation uses only 300-dim GloVe word embedding since ELMo, POS, and NER embeding require utilizing pre-trained weights in my model which is not permitted by project rules. The paper doesn't list the performance of Mnemonic Reader model on SQuAD 2.0. However, an open source implementation using TensorFlow[10] reported EM score of 64.89 and F1 score of 67.81 using ELMo and CoVe embedding.

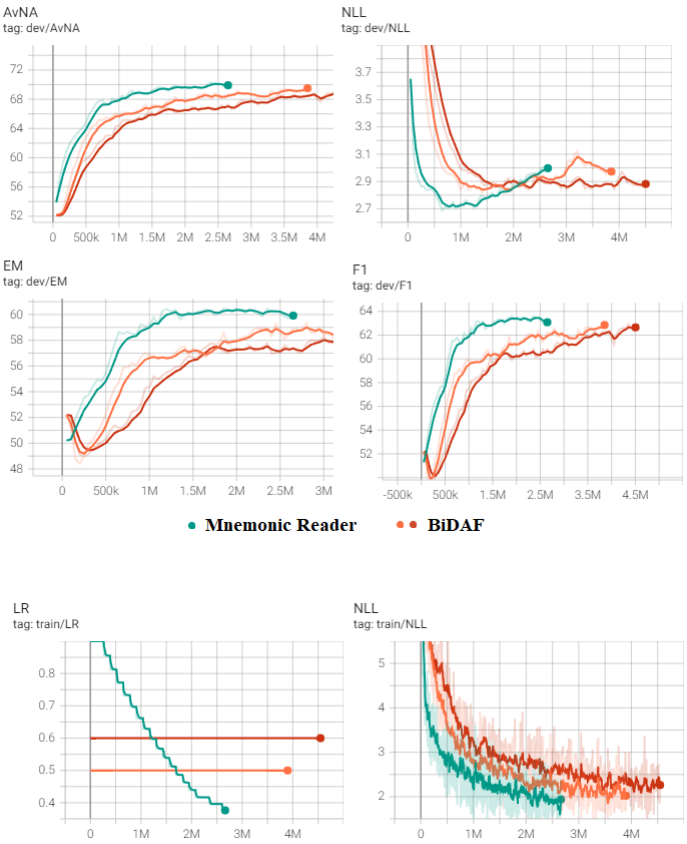

Figure 2: Training results of best performing models

Test Case I present a test case from the dev set where Mnemonic Reader model performed better than BiDAF baseline model.

Question: Name a larger car that Toyota came up with as buyers lamented the small sized compacts?

Context: Some buyers lamented the small size of the first Japanese compacts. Both Toyota and Nissan (then known as) introduced larger cars such as the Toyota Corona Mark II, the Mazda 616 and 810 , which added passenger space and a amenities such as air conditioning, power steering, AM - FM radios , and even power windows and central locking without increasing the price of the vehicle. A decade after the 1973 oil crisis, Honda, Toyota and Nissan, affected by the 1981 voluntary export restraints, opened US assembly plants and established their luxury divisions (Acura, and Infiniti, respectively) to distinguish themselves from their mass - market brands.

Answer:
Corona Mark II

Mnemonic Reader Answer:
Toyota Corona Mark II

BiDAF Answer:
Toyota Corona Mark II, the Toyota Cressida, the Mazda 616 and Datsun 810

Attention Concentration I present the attention concentration for the test case.

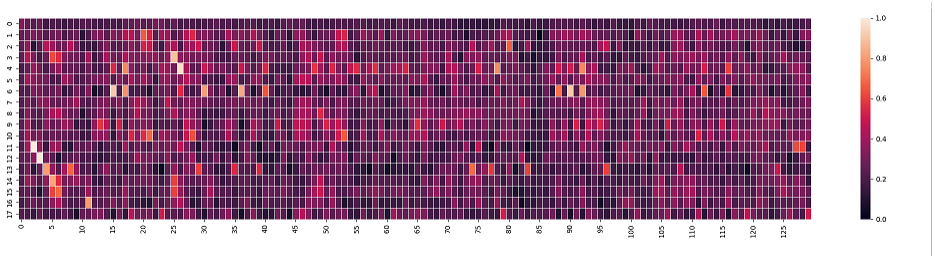

Figure 3: BiDAF Bi-Attention Score Matrix for the test case. X-axis indicate context words, and Y-axis axe indicate question words.

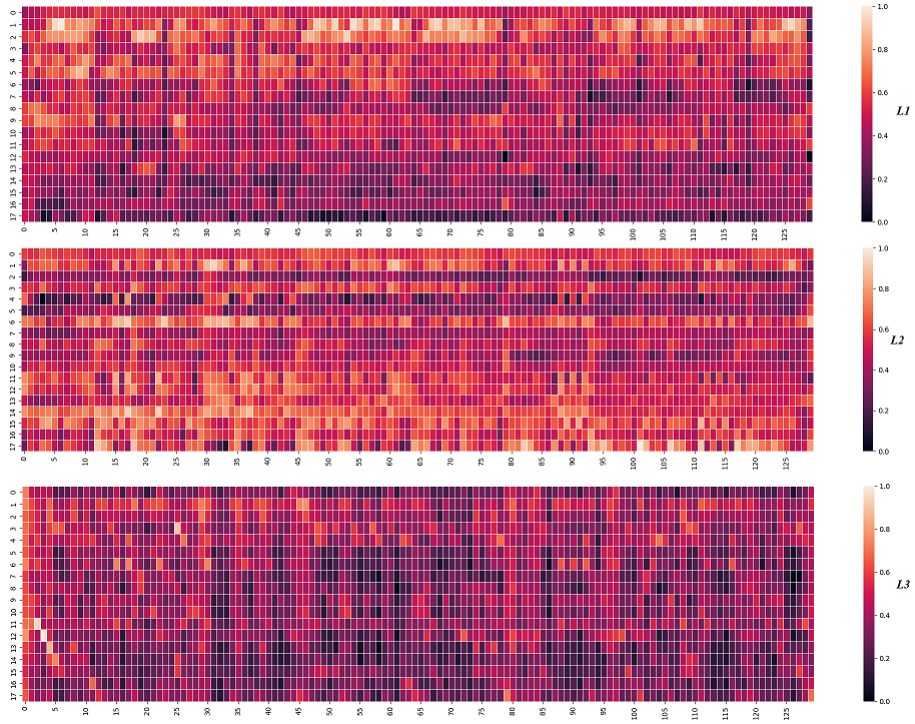

Figure 4: Mnemonic Reader 3-level Re-Attention Score Matrix for the test case. X-axis indicate context words, and Y-axis indicate question words.

The visual inspection of the attention matrices shows that BiDAF matrix is more concentrated on a few areas. The Mnemonic Reader attention matrix in the first layer seemed distracted but it seems to calibrate attention at each level. In the last layer, attention is significantly more concentrated at the intersection between "car" token in the question and "Corona Mark II" tokens in the context.

7 Conclusion

In this project I present an implementation of Mnemonic Reader that achieves better results on the SQUAD 2.0 dataset than the BiDAF baseline model. My experiments showed that increasing the hidden size beyond a certain level didn't improve the model performance. Also, my incremental addition of features e.g. normalized term frequency and exact match improved the model by 0.5 point on average. My experiment showed that re-attention was able to correct it's distraction in previous level and reach the correct information.

References

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

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

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

[4] 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.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[6] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.

[7] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[8] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*, 2018.

[9] 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.

[10] ewrfcas. Reinforced mnemonic reader in tensorflow. *GitHub, https://github.com/ewrfcas/Reinforced-Mnemonic-Reader*, 2018.