Improving Reasoning in Multi-Hop QA with Self-attention

Minsoo Kim
minsoo9574@gmail.com

Abstract

Single-hop question-answering architectures have been successfully extended to several multi-hop settings. However, it remains unclear to what extent they can facilitate complex, multi-step reasoning. To address this question, we propose two extensions to the standard QA framework, a memory-augmented retrieval module and a masked word-sentence bi-attention layer. These additions leverage self-attention and support sentence labels from the HotpotQA dataset to improve a model's performance on questions requiring complex reasoning. We show experimentally that our model meaningfully improves performance on both types of reasoning found in the questions of HotpotQA, and is particularly effective on comparison questions. Finally, we also show how analyzing support sentence predictions and answer predictions together can lead to better identification of explainability problems for QA architectures.

Query: What year did The Chronicle of Philanthropy's publishing overlap with Antic?
Type: Comparison
Answer: 1988
Supporting Sentence 1: Antic (ISSN 0113-1141) was a home computer magazine devoted to the Atari 8-bit family (Atari 400/800, XL, XE, XEGS).
Supporting Sentence 2: The magazine was published from April 1982 until June/July 1990.
Supporting sentence 3: The Chronicle of Philanthropy is a magazine that covers the nonprofit world.
Supporting sentence 4: It was founded in 1988 by editor Phil Semas and then managing editor Stacy Palmer.

Table 1: An example of a question from the HotpotQA dataset.

1 Introduction

Question answering has seen rapid advances in recent years due to a combination of large-scale datasets such as CNN/Daily Mail (Hermann et al., 2015), SQuAD (Rajpurkar et al., 2016) and advances in architectures (Hermann et al., 2015; Seo et al., 2017; Xiong et al., 2017). While the performance of these systems are impressive, it is questionable to what extent they can perform complex reasoning, a cornerstone of robust question answering. One issue is that the datasets themselves do not necessarily require complex reasoning to solve. For instance, Chen et al. (2016) and Min et al. (2018), find for CNN/Daily Mail and SQuAD, respectively, that many of the question included are often susceptible to relatively simple answering strategies, such as identifying a single evidence sentence, then applying lexical or paraphrase matching to find the answer. For data-driven systems, this poses a fundamental limitation on their ability to develop complex reasoning. Recent datasets have attempted to address these issues: TriviaQA (Joshi et al., 2017), NewsQA (Trischler et al., 2017) and SearchQA (Dunn et al., 2017) gather context or supporting documents post hoc, but as pointed out by Yang et al. (2018), still fall short of providing tasks that require truly complex reasoning to solve.

Yang et al. (2018) explicitly tackle this issue with a recent dataset, HotpotQA, by collecting questions that require finding and reasoning over multiple supporting documents to answer. Notably, HotpotQA also provides sentence-level supporting facts required for reasoning, as shown in Table 1, in addition to the answer labels. This type of supervision has not been considered in previous datasets dealing with large texts.

In this paper, we propose Self-attentive QA (SAQA), a model designed to take full advantage of the supporting fact supervision provided in HotpotQA, through two different applications of self-attention. SAQA augments the standard RNN and self-attention-based architecture with two new components. The first is a memory-augmented retrieval module (MRM) which learns to perform a joint support sentence selection & sentence relation modeling task. In MRM, a recurrent controller interacts with the retrieval targets, while a recurrent 2-D memory uses multi-head attention to model the potential interactions between the retrieved sentences. The second component is a masked word-sentence biattention layer, which uses the retrieval results from the MRM as an attention-focusing mask to relate sentence-level representations to their token-level counterparts. We perform our comparison against a strong baseline and show that the model leads to performance gains on all settings of the HotpotQA dataset. In particular, we show that it achieves greater improvement in the subset of questions which require comparative reasoning to solve.

Additionally, we find that utilizing the extra labels in HotpotQA has benefits beyond better question answering performance. Analyzing the support sentence selection module jointly with the question answering module allows us to check the consistency of the model's reasoning (i.e. is there positive correlation between support sentence identification and question answering?). Furthermore, by analyzing the failure cases where one module fails and one does not, we are also able to more accurately identify areas where improvements may be needed.

2 Model

In this section, we describe the overall SAQA model and the two central components, the memory augmented retrieval module and the masked word-sentence bi-attention layer.

2.1 Encoding Layers

Inputs to our model are first encoded using single-layer bi-directional RNNs', yielding sequence-aware representations of the context and query. Next, a bi-attention layer (Seo et al., 2017; Xiong et al., 2017) is applied to model the interactions between the context and the query. These representations are then fed into a stack of three residual self-attention layers with ReLU activations. As a whole, the layers yield the context token representations:

$$c_j = EncodingLayers(Token_j) \tag{1}$$

We adopt the formulation of bi-directional attention from (Clark and Gardner, 2018) as the basis for the context-query bi-attention, masked word-sentence bi-attention, and the self-attention layers. We adopt GRU (Cho et al., 2014) as the RNN throughout the model.

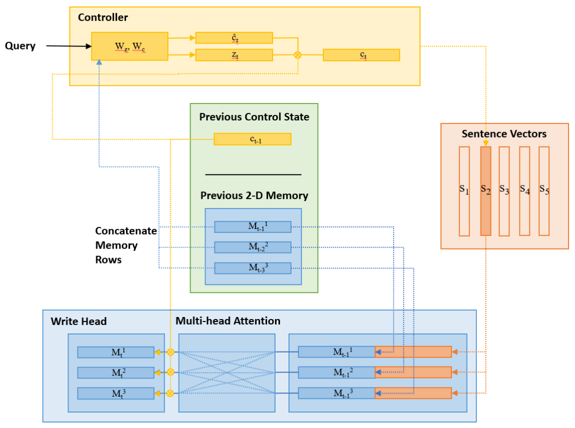

Figure 1: Operations of the memory-augmented retrieval module during a single timestep. The figure displays how the controller and the write head interact. The dotted lines represent inter-module operations while the full lines indicate intra-module operations.

2.2 Inputs to the Retrieval Module

Multiple strategies are possible to tackle the support sentence retrieval problem. Support sentence labels can be used as either sentence labels or token labels. The approach taken in Yang et al. (2018) applies another bi-directional RNN over c_j in Eq.(1) to create dedicated sentence-level vectors. We follow the same approach as it nicely reduces the scope of the problem to a selection/retrieval one, with the number of choices being bounded by the maximum number of sentences.

Let the output of the secondary bi-directional RNN be the concatenation of forward and backward states:

$$r_i = [h_i^{Forward}, h_i^{Backward}]$$

We obtain the set of sentence representations $\mathcal{S}$, by indexing these hidden states at the start and end indices of sentences:

$$s_i = [r_{iEND}^{Forward}, r_{iSTART}^{Backward}] \tag{2}$$

$$\mathcal{S} = \{s_1, \dots, s_n\} \tag{3}$$

2.3 Memory-augmented Retrieval Module

The goal of the memory-augmented retrieval module is to retrieve the correct support sentences, while letting each new retrieval inform the downstream QA task. To this end, we view the problem as a dynamic computing task involving separate control and write components (Graves et al., 2016; Hudson and Manning, 2018). This allows us to approach the task in a recurrent manner, retrieving one sentence at a time (control), and incorporating the retrieved sentence into a persistent memory (write). Overall, our approach is similar to Nishida et al. (2019), who use a single RNN in conjunction with a glimpse attention vector over the query tokens to retrieve sentences one at a time. While they approach the problem as a summarization task via sentence extraction, we find that there is a natural opportunity during retrieval to model further interactions in the sentence representation space. This is the reason for the more involved architecture of the MRM, which we will show to be meaningful in terms of performance.

Figure 1 shows the overall structure of the memory-augmented retrieval module. The operation of the module at each time-step t is defined as follows:

$$c_t = \text{Controller}(M_{t-1}, c_{t-1}) \tag{4}$$

$$r_t = \mathit{Retrieval}(c_t) \tag{5}$$

$$M_t = \text{WriteHead}(M_{t-1}, r_t) \tag{6}$$

Below, we describe each subcomponent in detail.

2.4 Controller

The interaction of the control state with the sentence vectors $\mathcal{S}$ determines the retrieval operation of the module. The controller is a gated recurrent network which receives the memory M_{t-1}, the encoded query q, and the previous control state c_{t-1} as input, and outputs the next control state c_t:

$$m_t = \mathit{Concat}(M_t)W_d \tag{7}$$

$$z_t = \sigma(W_z(m_{t-1} \odot q) + b_z) \tag{8}$$

$$\tilde{c}_t = \tanh(W_c[m_{t-1}, q] + b_c) \tag{9}$$

$$c_t = z_t \odot \tilde{c}_t + (1 - z_t) \odot c_{t-1} \tag{10}$$

The matrix W_d projects the concatenation of M_t's memory slots into a single vector m_t, then this memory is blended with the query q to perform a gated update of the recurrent control state, where $\odot$ denotes an element-wise product. The new control state c_t will define the subsequent retrieval operation.

2.5 Retrieval

The control vector interfaces with the sentence vectors to choose the next retrieval target. We compute inner products between c_t and the sentence vectors s_i to obtain an attention distribution over sentences:

$$\alpha_{ti}^{Read} = \text{softmax}(c_t \cdot s_i) \tag{11}$$

This attention distribution defines the probability, at timestep t, that the sentence s_i should be retrieved next. The retrieval output r_t is computed as follows:

$$r_t = \sum_i \alpha_{ti}^{Read} s_i \tag{12}$$

This allows the output to be a soft-attention weighted summary of the sentence vectors, or a hard selection of a single sentence vector. We opt for the latter, choosing s_i where $i = \text{argmax}_i\, \alpha_{ti}^{Read}$.

2.6 Write Head

With the retrieved sentence r_t, we update the memory matrix M_{t-1} by incorporating the newly obtained information. First, the retrieved sentence vector is combined with each row of M_{t-1} via multi-head attention:

$$Q_t = M_{t-1}W_q \tag{13}$$

$$K_t = [M_{t-1}; r_t]W_k \tag{14}$$

$$V_t = [M_{t-1}; r_t]W_v \tag{15}$$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{16}$$

Eq.(16) defines a single attention *head*, utilizing query-key-value projections and a scaled dot product, where d_k is the projection dimensionality. In multi-head attention, these projections are performed multiple times, making it possible to attend to different subspaces of the representation at hand. Since sentence representations are composites of word representations, we posit that multi-head attention will be useful in focusing on different specific parts of the sentences such as entities, dates, etc. The results of the different attention heads are concatenated and projected into a single output:

$$\text{MHA}(Q, K, V) = [head1; ...; head_h]W_o \tag{17}$$

$$\tilde{M_t} = \text{MHA}(Q_t, K_t, V_t) \tag{18}$$

Finally, we perform a row-wise recurrent update on the memory matrix, by computing a gating function dependent on the context vector and each memory row. The projection weights W_z are shared.

$$Z_t = \sigma(W_z(\tilde{M}_t \odot c_{t-1}) + b_z) \quad (19)$$

$$M_t = Z_t \odot M_{t-1} + (1 - Z_{t-1}) \odot \tilde{M}_t \quad (20)$$

2.7 Word-Sentence Bi-attention Layer with Support Masking

Since the retrieval task necessitates dealing with vectors in the sentence representation space, we utilize self-attention as a way to integrate sentence-level representations with token-level representations.

We compute a bi-attention between the context sentence vectors s_i and context token vectors c_j.

$$a_{ij} = W_1 \cdot s_i + W_2 \cdot c_j + W_3 \cdot (s_i \odot c_j) \quad (21)$$

Here, we modify the bi-attention by using the support sentences selected by MRM. We apply a constraint on the j dimension of a_{ij} by masking out the context token indices which do not belong to the selected support sentences:

$$a_{ij} = a_{ij} * Mask_j \quad (22)$$

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_c} e^{a_{ij}}} \quad (23)$$

We then compute vectors S_i corresponding to each sentence as:

$$S_i = \sum_{j=1}^{n_c} c_j p_{ij} \quad (24)$$

Since p_{ij} is masked, the weighted sum involves only the tokens in support sentences. In effect, S_i is a view of the set of support sentence tokens with an awareness of how they relate to sentence s_i. For clarification, we do not model the relation between *support* sentences and support sentences tokens, but between *all* sentences and support sentence tokens.

We also compute the complimentary word-to-sentence vector C_s, which is a weighted sum of sentence vectors, where the attention focuses on important words in the support sentences:

$$m_i = \max_{1 \le j \le n_c} a_{ij} \quad (25)$$

$$p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}} \quad (26)$$

$$C_s = \sum_{i=1}^{n_s} s_i p_i \quad (27)$$

The final output of the layer is a concatenation of s_i, S_i, $s_i \odot S_i$, and $C_s \odot S_i$. Similar to Yang et al. (2018), the output vectors are fed as additional inputs to the final span prediction layer along with context vectors c_j.

3 Training

3.1 Loss functions

During training, we provide direct supervision on Eq.(11) via a cross-entropy loss between α_t^{Read} and the correct support sentence distribution $\hat{\alpha_t}$. The target distribution $\hat{\alpha_t}$ is given for each example by setting one of the support sentence as the target for a timestep t. We use teacher forcing, and randomly shuffle the order of the labels at every iteration. Trained this way, the model is agnostic to the order that sentences are retrieved. Additionally, we also include a learnable vector in the set of sentence vectors, which is analogous to the END token. This allows the module to terminate and stop retrieving more sentences as it sees fit. Overall, the loss for the support sentence retrieval task is:

$$L_{Support} = \sum_t \sum_i \hat{\alpha_{ti}} \log(\alpha_{ti}^{Read})$$

In our examination of the data, we found that often, the correct answer span occurs multiple times in the text. In particular, such duplicate answer spans occur frequently even when only considering the support sentences. Therefore, we define a span prediction loss that averages over all mentions of the answer span within support sentences:

$$L_{Answer} = -\frac{1}{N_M} \sum_i^{N_M} \log(\mathbf{p}^{start}_{y_i^{start}}) + \log(\mathbf{p}^{end}_{y_i^{end}})$$

We train the network with a final loss term that sums the answer and support prediction losses.

$$L = L_{Answer} + \lambda L_{Support}$$

We set $\lambda = 1$ in all of our experiments.

3.2 Input features

We use pre-trained GloVe (Pennington et al., 2014) word and character embeddings, as well as pre-trained BERT (Devlin et al., 2018) embeddings, for all context and question text. We tokenize text using Spacy (Honnibal and Montani, 2017). All features are concatenated together before being fed into the model. For BERT embeddings, since our input is a large text consisting of

Model	Answer		Sup Fact		Joint	
	EM	F1	EM	F1	EM	F1
QFE (Nishida et al., 2019)	53.86	68.06	57.75	84.49	34.63	59.61
DecompRC (Min et al., 2019)	55.20	69.63	N/A	N/A	N/A	N/A
DFGN (Qiu et al., 2019)	56.31	69.69	51.50	81.62	33.62	59.82
Ours	55.07	70.22	57.62	84.19	35.94	61.72
TAP2 (ensemble)**	66.64	79.82	57.21	86.69	41.21	70.65
QFE (Nishida et al., 2019)	28.66	38.06	14.20	44.35	8.69	23.10
Ours	28.44	38.62	14.69	47.17	8.62	24.49
MUPPET (Feldman and El-Yaniv, 2019)	30.61	40.26	16.65	47.33	10.85	27.01
DecompRC (Min et al., 2019)	30.00	40.65	N/A	N/A	N/A	N/A
Cognitive Graph QA (Ding et al., 2019)	37.12	48.87	22.82	57.69	12.42	34.92
BERT pip.**	45.32	57.34	38.67	70.83	25.14	47.60

Table 2: Results on the test sets of HotpotQA, in the distractor and fullwiki settings. Due to the leaderboard being large, We only list the published models at the time of writing, but also include the current state-of-the-art[2]. For each setting, the best unpublished model is denoted by **.

10 concatenated paragraphs, the maximum number of tokens well exceeds the BERT maximum of 512 words. Therefore, we use a sliding window of 512 words with a stride of 256 words to cover the entire context, averaging the overlaps. Furthermore, because BERT uses a wordpiece vocabulary, we map the wordpiece output vectors to their corresponding words via averaging. We use the base cased BERT model provided by the authors, extracting vectors from the penultimate layer.

3.3 Optimization

We train the network for 15 epochs using Adam optimizer, with cosine annealing (Loshchilov and Hutter, 2017) of the learning rate from 0.001 to 0.0001. We found learning rate annealing to have a significant impact on performance, and tested between cosine annealing and plateau annealing based on evaluation F1 on the development set. When using BERT embeddings, cosine annealing was better, while when using GloVe alone, plateau annealing performed best. For the latter we used an initial learning rate of 0.001, patience of 2, and decay factor of 0.5. We used two Nvidia GeForce GTX 1080 Ti GPUs'[1], and the batch size for all experiments was set to 24. We evaluated on the development set every 1000 iterations, and the model with the highest validation F1 score was selected as the final model. We found the model to saturate at around 11 epochs.

4 Results

We report our results on the HotpotQA test set in Table 2. HotpotQA evaluates three sets of scores: The answer span prediction score, the support sentence prediction score, and the joint score of the two. Furthermore, there are two settings, distractor and fullwiki. The distractor setting includes 2 gold paragraphs with 8 distractor paragraphs, while the fullwiki setting does not include gold paragraphs, and instead, all paragraphs are retrieved from Wikipedia based on tf-idf similarity with the query. Since we do not focus on the open-domain paragraph retrieval problem tested in the fullwiki setting, we focus our analysis on the development setting. However, we include results in both settings to show that our model is robust to noisy inputs. Our model is on par with similar competitive models, including all of the published models, on the distractor setting. However, we also note that unpublished models on the leaderboard now far outperform the currently published ones. Since the unpublished models do not specifiy details such as paragraph-wise processing, usage of more recent/powerful pre-trained embeddings, etc., we focus our analysis on our own, directly comparable strong baseline. For completeness, we list the current best model on the leaderboard as well.

[1]Code for our model and experiments will be released.

Model	EM	F1
SAQA + BERT	55.43	70.62
Base + BERT (Yang et al., 2018)*	54.35	69.67
SAQA	51.19	66.48
Base (Yang et al., 2018)*	50.30	64.80

Table 3: Improvements of SAQA over the baseline. * denotes our implementation.

4.1 Does SAQA scale with BERT embeddings?

In Table 3, we report the performance of the base model and SAQA, with and without BERT embeddings. We find that SAQA performs consistently better than the base model in both settings. However, the gap between the models decreases from 1.68 F1 to 0.95 F1 when using BERT embeddings. This is due to the noticeable jump in the performance of the base model. Overall, SAQA constitutes a meaningful increase over a strong baseline that leverages a modified loss function, fine-tuned optimization, and both GloVe and BERT embeddings.

[2]The full leaderboard of HotpotQA can be found at https://hotpotqa.github.io/

Model	EM	F1
Masked word-sentence bi-att + 2 memory slots, 8 heads	55.43	70.62
No masked word-sentence bi-att	54.71	70.18
2 memory slots, 4 heads	54.99	70.37
3 memory slots, 8 heads	55.09	70.49
No write head	54.62	70.13

Table 4: Effect of components on prediction performance (dev set, distractor setting).

Model	Type	Ans F1	Sup Rec.	Sup Prec.
SAQA + BERT	Comparison	65.92	89.67	92.71
Base + BERT	Comparison	63.32	85.72	88.39
SAQA + BERT	Bridge	71.81	79.60	87.89
Base + BERT	Bridge	71.26	74.05	84.62

Table 5: Performance breakdown by question type.

4.2 Ablation study

We validate our design choices by ablating each component in Table 4. The top block shows the full model, which uses both the MRM with write head and the masked bi-attention, while each block below represents the ablation of a single component from the full model. First, the ”No write head” entry modifies MRM so that the write head is removed, along with the 2-D memory, leaving only the single recurrence of the controller. In this case the performance drops by 0.49 F1. We also find that the hyperparameters of the 2-D memory have an impact on the performance, with 2 memory slots + 8 heads performing best, but overall, regardless of hyperparameters, it is always beneficial to include the write head with 2-D memory. The ”No masked word-sentence bi-att” entry removes the masked bi-attention from the full model. This component also contributes to a 0.44 F1 increase in performance. Since the overall increase from the combination of our model’s two components over the baseline is 0.95 F1, we can conclude that they are complimentary.

4.3 Performance by question type

In Table 5, we break down the performance of our model by question type. There are two main question types in HotpotQA, bridge and comparison. Bridge questions require identifying multiple entities over multiple hops to either a) infer bridge entities given a set of properties, b) check multiple properties of an entity or c) infer about the property of another entity through the bridge entity. Comparison questions, on the other hand, require the system to compare specific properties of two entities such as age, number of years in office, etc. In particular, text-based comparison questions are a novel type of question that has not been considered in previous datasets. These questions pose some challenges for standard models: They need to be solved by performing relational reasoning about the properties of two entities, and these properties often require numeracy and counting to resolve (Which band has more albums than the other?). There are also questions which require some grounded knowledge about geography (Which city is farther West than Tokyo?), or time (i.e. inferring the ages of entities from their birthdates).

In Table 5, we find that our model outperforms the base model by 2.6 F1, and 0.55 F1 on comparison and bridge questions, respectively, which confirms that our model improves reasoning capability for both types of questions, but especially so on comparison questions. We note that there are 1487 and 5918 comparison and bridge questions in the development set, respectively, at a ratio of 1 to 4. Because there are fewer comparison questions, the effect on performance is less pronounced when aggregating the question types together.

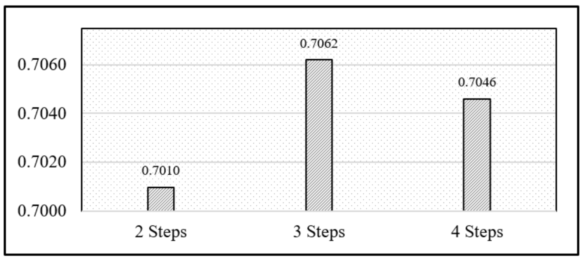

Figure 2: Performance effect of the maximum number of steps for the MRM.

5 Analysis

5.1 How does SAQA handle multi-hop questions?

HotpotQA consists of multi-hop questions provided along with their gold support sentences, which roughly correspond to each hop. Because we retrieve one such sentence at a time, setting the maximum number of retrieval steps allows us to set the maximum number of hops, or reasoning steps. We tuned this number as a hyperparameter, and found that setting the maximum number of steps for the MRM to 3 led to the best performance. Figure 2 shows the effect of varying the numbers of maximum steps on question-answering performance. Because of the random shuffling of support sentence labels, setting the maximum steps to a lower number does not prevent the model from utilizing all of the labels. This suggests that there is an optimal number of reasoning steps for HotpotQA, at 3 steps. We find that this is consistent with our observation of the dataset. There are 4990, 1774, 537, and 104 questions corresponding to 2,3,4, or greater than 4 support sentences, with 2 or 3 step questions making up the majority. We also compare the base model and SAQA by their performance on questions with different numbers of gold supports in Fig. 3. As expected, SAQA outperforms the base model for all cases except for questions with 4 gold supports, where the base model performs marginally better. These results suggest that our model improves multi-hop reasoning capability generally, and the performance gains are not limited to some subset of questions.

5.2 Relationship between sentence retrieval and question answering

We also study the relationship between the retrieval task and the question answering task to better understand the properties of our model. We begin by noting that in Table 5, there is a negative correlation between support sentence retrieval performance and question answering performance, when comparing across the two question types. That is, the model identifies the correct support sentences for comparison questions more easily, but answer bridge questions more easily. For bridge questions, both recall and precision on the support sentence retrieval task is higher relative to comparison questions by 5 to 10 points but the answer F1 is lower by more than 6 points. On the one hand, it is not so surprising that support sentence retrieval is easier for comparison questions: Comparison questions are more easily identified by finding exact matches of entities whereas bridge questions will often require co-reference resolution for the bridging step. On the other hand, it is less clear why answering comparison questions is harder. Likewise, Min et al. (2019); Nishida et al. (2019) report these discrepancies between the bridge and comparison questions.

To better understand this finding, we subdivide all of the development set predictions into four buckets based on the support retrieval F1 score in Table 6. First, we note that there is a positive correlation between the support F1 and answer F1 scores, indicating that the model utilizes the support predictions to inform its answer. Next, we focus our attention on the final bucket corresponding to the 75 to 100 support sentence retrieval F1 range. We choose this bucket for two reasons. First, it accounts for the majority of all predictions (74% for bridge questions, 83% for comparison questions). Second, qualitatively, the predictions included in this bucket represent satisfactory retrieval, allowing us to look at question answering performance in isolation. The average F1 scores for this bucket are 68.5 and 78.3, for comparison and bridge questions, respectively. That is, when retrieval is satisfactory, the model is noticeably worse at answering comparison questions.

Similarly, we also identify the predictions where the model has done a perfect job on support sentence retrieval (support F1 is 100), and count the subset of these where the answer F1 score is less than or equal to 50. These represent the most extreme mismatch cases between support sentence retrieval and question answering performance, and in our view, are the most problematic. We observed the questions belonging to this subset to be trivially easy for humans to answer, if the correct support sentences are found. Table 7 displays the sizes of these subsets, for each question type. The percentage values show the proportion of such cases out of all questions of a given type.

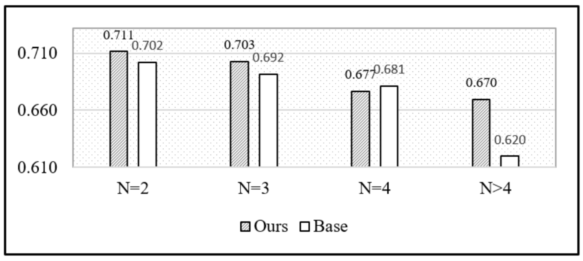

Figure 3: Comparison between SAQA and the base model based on the number of gold support sentences.

Support F1 Range	Comparison		Bridge	
	Avg. F1	Ratio	Avg. F1	Ratio
0, 25	22.22	0.8%	24.3	1.2 %
25,50	56.12	3.1%	48.74	5.6%
50,75	54.13	13.5%	57.56	18.2%
75,100	68.49	82.6%	78.28	74.4%

Table 6: Development set predictions categorized into buckets of support F1 score ranges.

Type	Answer F1	Support F1	N	Ratio
Comparison	<= 50	== 100	334	22.46 %
Comparison	All	== 100	1112	74.78%
Bridge	<= 50	== 100	643	10.86 %
Bridge	All	== 100	3349	56.59%

Table 7: Extreme mismatch cases where support sentence retrieval is perfect, but answer prediction is poor

Query: Who were the directors of the 2009 American science fiction film starring the actor who played Dexter Morgan in the Showtime TV Network series " Dexter " ?
Type: Bridge
Answer: Mark Neveldine and Brian Taylor
Sentence 1: Gamer is a 2009 American science fiction action film written and directed by Mark Neveldine and Brian Taylor.
Sentence 2: Michael Carlyle Hall (born February 1 , 1971) is an American actor , known for his roles as Dexter Morgan , a serial killer and blood spatter analyst , in the Showtime TV Network series " Dexter " , and as David Fisher in the HBO drama series " Six Feet Under " .
Sentence 3: Alongside Butler and Lerman , it also stars Michael C. Hall , Ludacris , Amber Valletta , Terry Crews , Alison Lohman , John Leguizamo , and Zo Bell .
Sentence 4: The film stars Gerard Butler as a participant in an online game in which participants can control human beings as players , and Logan Lerman as the player who controls him .

Step	1		2		3
Sentence 1 Score	$\underline{0.4777}$	$\longrightarrow$	0	$\longrightarrow$	0
Sentence 2 Score	0.2415	$\longrightarrow$	0.4465	$\longrightarrow$	$\underline{0.9144}$
Sentence 3 Score	0.2305	$\longrightarrow$	$\underline{0.4859}$	$\longrightarrow$	0.062
Sentence 4 Score	0.05	$\longrightarrow$	0.0636	$\longrightarrow$	0.0143

Table 8: An illustration of SAQA's support sentence retrieval and question answering process.

We find that for comparison questions, this rate amounts to a significant, 22.5% of all comparison questions. For bridge questions, the same rate is nearly halved, at 10.9% of all bridge questions. In summary, we find that comparison questions constitute a harder challenge for current models, both at the aggregate level and in terms of the most extreme cases. In light of this, the fact that our model achieves a significant improvement on comparison questions is encouraging, but more work is needed on understanding how and why these disparities arise between reasoning sub-types.

Finally, with these cases, we hope to demonstrate the value of examining intermediate reasoning processes for the goal of building explainable question answering systems, rather than relying only on the final question-answering performance metric. From the standpoint of human reasoning, it is difficult to justify the failure to deduce an answer when already having correctly identified the requisite supporting facts. A bona-fide explainable question answering system should aim to minimize such mismatch between support sentence identification and question answering.

5.3 Qualitative Analysis

We provide an example of our model's sequential support sentence selection and question answering capability in Table 8. The bottom of the table shows how the retrieval scores for the top four sentences change over time. At each time step, the model selects the highest scoring sentence for retrieval, indicated by the underline. After retrieving three sentences, the model selects the correct answer span, informed by the retrievals.

6 Conclusion

We introduce Self-attentive QA, which proposes augmenting question-answering architectures with two different applications of self-attention, the memory-augmented retrieval module and the masked word-sentence bi-attention layer, to better facilitate reasoning on complex, multi-hop questions. By leveraging support sentence labels in question answering, our model improves performance against a strong baseline on both types of reasoning found in the questions of HotpotQA, especially on comparison questions. Analyzing the two modules together also enables us to more clearly identify areas where current models may have weaknesses, contributing to the overall explainability of the model.

References

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855. Association for Computational Linguistics.

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

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, Florence, Italy. Association for Computational Linguistics.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2309, Florence, Italy. Association for Computational Linguistics.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwiska, Sergio Gmez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adri Puigdomnech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611. Association for Computational Linguistics.

I. Loshchilov and F. Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1725–1735. Association for Computational Linguistics.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2335–2345, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150, Florence, Italy. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200. Association for Computational Linguistics.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *International Conference on Learning Representations*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics.