Answering Reading Comprehension Using Memory Networks

Darshan Kapashi
Department of Computer Science
Stanford University
Stanford, CA 94305
darshank@stanford.edu

Pararth Shah
Department of Computer Science
Stanford University
Stanford, CA 94305
pararth@stanford.edu

Abstract

In this work, we will investigate the task of building a Question Answering system using deep neural networks augmented with a memory component. Our goal is to implement the MemNN and its extensions described in [10] and [8] and apply it on the bAbI QA tasks introduced in [9]. Unlike simulated datasets like bAbI, the vanilla MemNN system is not sufficient to achieve satisfactory performance on real-world QA datasets like Wiki QA [6] and MCTest [5]. We will explore extensions to the proposed MemNN systems to make it work on these complex datasets.

1 Introduction

A long term goal of NLP has been to develop a general purpose AI agent which can hold a natural language dialog with a human participant. But it is difficult to automatically evaluate the performance of an agent in general dialogue, which makes it hard to devise a learning method for improving the agent's performance. However, the task of question answering (QA) fits this criteria easily since the agent's response to a question can be evaluated against the expected answer. Additionally, QA is extremely broad as many NLP tasks can be reformulated in the QA setup. This implies that devising better models for improving accuracy and efficiency of agents on QA tasks can be quite useful.

Fundamentally, QA systems must perform two tasks: retrieval and inference. The QA system must store the knowledge presented to it in some convenient internal representation and subsequently, it must search through this knowledge bank and retrieve the pieces of information that would help to answer the question posed to the system. Analyzing the question text in this process requires some form of inference, which can be done either via explicit logical rules, e.g. predicate calculus to infer what is being asked for, or it could be more implicit, e.g. via the trained network parameters of a sequence predictor like RNN or LSTM model. Both the tasks, retrieval and inference, are critical to the success of a QA system.

In this work, we will introduce simple extensions to the Memory Networks framework which are aimed towards improving performance on specific QA tasks. The next section contains a brief description of related work in this area. Section 3 contains a formal description of the QA tasks under consideration. Section 4 provides a brief overview of the components of Memory Networks and details of our extensions to the same. Section 5 will present some results from our experiments and finally in Section 6 we discuss conclusions and future work.

2 Background

One can view early approaches to Artificial Intelligence (AI) and modern approaches to Machine Learning (ML) as extremes lying on a spectrum of methods [1]. Early AI used rules and ontologies, but these were heuristic, brittle, and non-scalable. Although ML approaches have been extended in various ways to handle structured data, the main underlying approach is predominantly statistical: the ML supervised setting usually considers a family of models, a cost function, and a labeled dataset, and the goal is to find that model for which the error measure at hand is minimized on out-of-sample data.

However, the two main threads of work on QA systems until now have not been successful in merging a large supervised memory with powerful inference models. The work in Fader et al [2] is representative of traditional QA systems, which induce a machine learning objective function that maps open-domain questions to queries over a database of web extractions, using handcrafted features that take into consideration lexical and syntactic patterns occurring in the question text as well as the knowledge bank. In contrast to this, we have recently seen that deep neural networks that use memory units like RNNs and LSTMs, are powerful sequence predictors that can be efficiently trained to learn to do inference over long term dependencies in the text. Where they fall short is in their lack of a structured memory component, which can simplify the inference task by allowing the network to focus on only the relevant pieces of information while answering the question.

Memory Networks (Weston et al [10]) is a recent model that aims to learn how to reason with inference components and a long-term memory component. Its memory serves as a knowledge base to recall facts from the past. Similar approaches for combining deep networks with a memory component for other tasks have also been published recently (Graves et al [3]). For the task of QA specifically, the model tries to learn a scoring function to rank relevant memories. At prediction time, the model finds k relevant memories according to the scoring function and conditions its output based on these. The hope is that even though LSTM may perform poorly on a complex QA task, LSTMs conditioned on the relevant memories will be much more effective.

3 Problem Statement

At a high level, the goal is to build an agent that can answer questions posed by humans in natural language. The task consists of reading a piece of text, which may be sentences forming a story, or a set of facts forming a knowledge base. Then, certain questions are asked based on the given text. The agent is expected to read the question and output an answer which may be a single word or a natural language sentence (we consider both as separate tasks).

Mary moved to the bathroom. John went to the hallway. Daniel went back to the hallway. Sandra moved to the garden.
Q: Where is Mary? A: bathroom
Q: Where is Daniel? A: hallway

Figure 1: A sample QA task which consists of 4 sentences of text and 2 questions based on the text.

A sample task is shown in Figure 1. The text is generated from a simulation consisting of a few actors, objects and places. The questions in this task are of basic factoid type with a single supporting fact, since the answer depends on information from a single sentence of text.

Weston et al [9] have presented a total of 20 such QA tasks of varying difficulty levels, each of which test different memory and inference skills of the AI agent. The tasks are designed to cover a broad set of linguistic comprehension skills, including factoid QA, negation, counting, coreference, conjunction, induction, deduction, positional reasoning, path finding, etc. We will not reproduce here all the tasks from the paper, but encourage the reader to follow the original paper for a detailed description.

3.1 Dataset

3.1.1 bAbI QA Tasks

The data for each task described in [9] is generated from simulations, similar to the one presented above. The authors have shared a standardized dataset consisting of 1000 training and 1000 test questions for each of the 20 tasks[1]. This is presented as a standard benchmark against which memory-based QA models can be tested for comparison of performance.

3.1.2 MCTest

As described by the authors, MCTest is a freely available set of stories and associated questions intended for research on the machine comprehension of text. Each story is a series of sentences followed by 4 multiple choice questions each having 4 options. An example story is shown in 2

> One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.
> Q: Where did James go after he went to the grocery store?
> (A) his deck (B) his freezer (C) a fast food restaurant (D) his room

Figure 2: A sample MCTest comprehension which consists of a paragraph of text and multiple-choice questions based on that text.

The MCTest dataset differs from the bAbI QA dataset in the following ways:

1. MCTest is open domain but restricted to concepts understandable by a 7 year old [6]
2. MCTest uses real natural language fictional stories unlike bAbI which has simulated data
3. Stories have up to 60 sentences and the vocabulary is $\sim$2000 words (in MC160) and $\sim$4000 words (in MC500). In comparison bAbI has $\sim$15 sentences and a vocabulary of $\sim$40 words

3.1.3 Wiki QA

This dataset has a corpus of Wikipedia articles and manually generated factoid-based questions and answers. It's corpus consists of real Wikipedia articles, each of which have roughly 100 to 2000 sentences, and a vocabulary of 40000+ words. The factoid questions are open-ended and not multiple choice, and the language complexity in this task is much higher than bAbI or MCTest, so unsurprisingly this dataset is much harder to perform well on.

4 Technical Approach

4.1 Supervised MemNNs

Supervised Memory Networks (or MemNNs) are designed to solve this exact problem of combining a memory component with an LSTM model for inference. We will follow the model described in [10], and direct the reader to the original work for a detailed description of Memory Networks. As a brief overview, we present the MemNN architecture in Figure 3. MemNN is divided into four components:

1. **Input** converts incoming input text to the internal feature representation.
2. **Generalization** takes the input feature vector and current memory and decides which slot to store the new input into. It can also modify/delete any earlier memory based on this new information, which can be seen as generalizing the stored knowledge as new pieces of information are encountered.

[1] http://fb.ai/babi

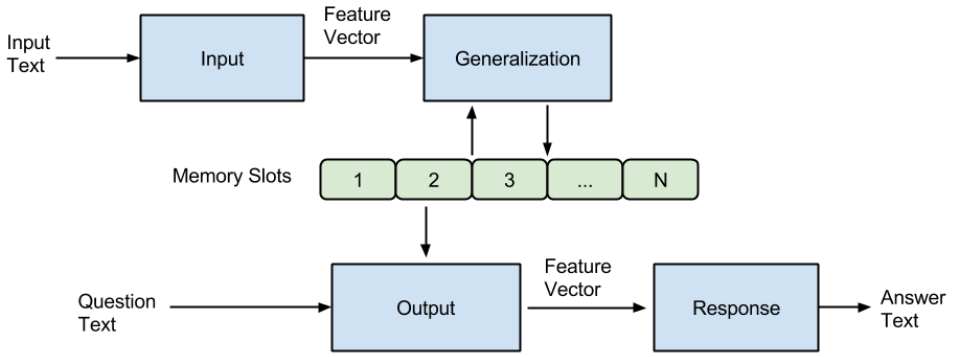

Figure 3: Memory Network architecture.

3. **Output** takes the question feature vector and current memory and generates a feature vector for the answer. This is where the inference must take place. In the simplest case, this can be implemented as a ranking function over all occupied memory slots, and the highest scoring supporting memory can be retrieved as:

$$o_1 = O_1(x, M) = argmax_i \; s_O(x, m_i)$$

where s_O is a function that scores the match between a question and the contents of a memory slot.

4. **Response** takes the answer feature vector and generates a natural language statement, which is outputted by the system. Ideally, an RNN or LSTM network that outputs a sequence of text tokens should suffice for this component.

The core innovation in MemNNs lies in formulating read/write operations to a memory as a differentiable function, thus allowing it to be trained via gradient descent with the rest of the neural network. This is similar in spirit to the parallely published work on Neural Turing Machines (Graves et al [3]). MemNNs effectively break down the QA task into two steps: finding the most relevant pieces of memory for the given question, and then using those memories to generate a natural language answer, while both components can be trained together under a common loss function. As we will see in Section 4, this two-step approach can greatly improve the performance of LSTMs when the article text increases in length.

4.2 Weakly Supervised MemNN

The memory network described in [10] needs the supporting memories for a given question in the training data. This in itself is prohibitive to applying the MemNN on other general QA tasks because each dataset will need to be annotated with the supporting statements for each question. [8] described another version of the MemNN which does not require this information. The framework is as described in [8], and we direct the reader to the original work for more details. We present a brief overview below.

Figure 4 shows 1 layer of the proposed network. There are 3 aspects to it:

1. **Input side:** Let the input sentence be

$$x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$$

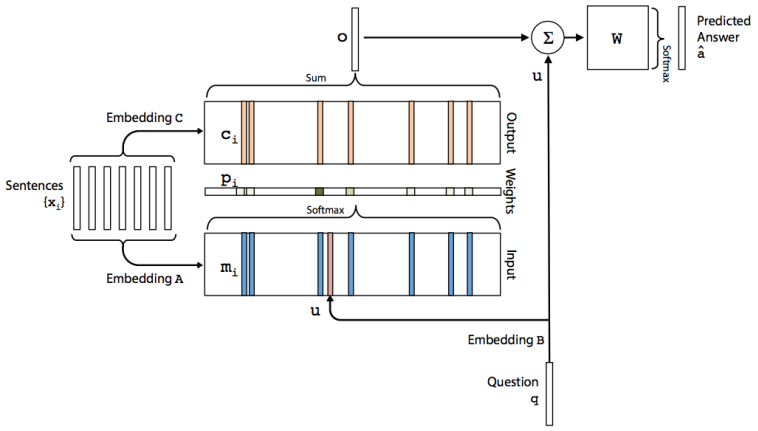

Figure 4: Weakly Supervised Memory Network

Each word of the sentence is embedded using a matrix A. For each sentence, we compute a corresponding memory m_i by summing up the embeddings of all the words in the sentence.

$$m_i = \sum_j Ax_{ij}$$

The question is embedded using another matrix B and we calculate the match between the question and each memory as

$$p_i = softmax(u^T m_i) = softmax(q^T B^T \sum_j Ax_{ij}).$$

2. **Output side:** Each memory vector has a corresponding output vector c_i, computed using another embedding matrix C

$$c_i = \sum_j Cx_{ij}$$

The output vector from the memory o is then a sum over the c_i, weighted by the probability vector from the input:

$$o = \sum_i p_i c_i$$

3. **Answer prediction:**

$$a = softmax(W(o + u))$$

One such layer corresponds to one memory lookup. We can stack many such layers and use $u^{k+1} = u^k + o^k$ and finally use the W matrix to predict an answer.

The model in [3] needs the training dataset to be annotated with supporting memories. Most datasets do not have this information. We applied the weakly supervised memory network to the Wiki QA and the MCTest dataset.

Next, we will describe our extensions to MemNN for the two real-world QA datasets.

4.3 Extensions for the Wiki QA Task

The memory network described in [10] and [8] trains and performs well on datasets with a small vocabulary. However, when the vocabulary size increases beyond the 40 words in the bAbI tasks, the model takes too long for a single epoch. We describe 2 methods we used to train the weakly supervised memory network on the Wiki QA dataset [9]. Both the approaches use some semantics of the story sentences to prune the set of statements for each question. Pruning (a form of pre-processing) helps in 2 ways. The easy to see advantage is in training the model. Applying a memory network to a real world dataset like the Wiki takes a long time. Secondly, it takes away noisy statements which helps the model find the relevant memories more effectively.

1. **WordVec based:** We use pre-trained word vectors. For each (question, statement) pair, we take the word vectors for each token and compute the cosine similarity between the question and the statement. We now have a score for each statement and can prune sentences below a certain threshold or take the top N statements. In a dataset with 100s of statements per question, this gives a huge boost in the model training speed.
2. **Part of speech (POS) based:** For each token in the question and each token in the story sentence, we find its POS tag. We prune statements which do not contain any nouns common with the question. This is one strategy. Depending on the dataset, we can choose a more relevant strategy and make the model train quicker.

4.4 Extensions for the MCTest Task

The Weakly Supervised Memory Network [8] is designed to output a probability distribution over the vocabulary. For the MCTest dataset, we are given 4 options for each question. One challenge is to be able to handle answers with multiple words. Another challenge (improvement) is to use the 4 given options to predict better.

To incorporate the options in the training process, we use a scoring function of the form:

$$p_i = u_k^T U^T U a_i$$

where

$$u_k = o_{k-1} + u_{k-1}$$

and a_i is the i-th option. We take the option with the maximum score as the predicted answer. During training we use the cross-entropy loss function. [2]

The advantage of such a scoring function is that the options are now a part of the training and we backpropagate on its errors. For an option of the form:

$$x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$$

Then we compute its embedding as:

$$a_i = \sum_j D x_{ij}$$

where D is the embedding matrix for the option.

This allows us to score and rank options with multiple words too.

5 Experiments

5.1 bAbI QA Tasks

As a first experiment, we trained an LSTM network which reads the paragraph text and a question, and outputs an answer word. The LSTM network performs poorly when it is fed the entire paragraph followed by the question[3], however, when it is fed just the sentences that are relevant to the question, it performs perfectly with 100% accuracy on the test set for Tasks QA1, QA2 and QA3 (See Table 1). This justifies the importance of using a memory component for QA tasks as done in MemNNs.

We used an embedding size of 100 for word vectors, and 100 for output of LSTM unit, which feeds to a dense layer followed by softmax activation. We trained the network using RMSProp to minimize the cross-entropy loss, and backpropagating through time from the answer word and the final output of the LSTM. We trained the network for 30 epochs.

We implemented the supervised MemNN network as described in [10], using bag of words representation for the input statements and question text. We also implemented write-time modeling (Section 3.4 of [10]) to take into account the relative order in which statements are written to memory. The

[2] We take the softmax of the p vector and use this as the score

[3] Our baseline numbers do not exactly match those reported in the original work because we did not perform task-specific hyperparameter tuning. We used the same model across all tasks to have our results be comparable across different tasks and input configurations.

Table 1: Test accuracies on bAbI QA tasks.

bAbI QA Task	LSTM w/ rel. stmts	LSTM w/ entire article	Supervised MemNN (k = 2)	Weakly sup. MemNN
QA1: Single Supporting Fact	100%	31.2%	100%	66.7%
QA2: Two Supporting Facts	100%	35.6%	77.5%	60.3%
QA3: Three Supporting Facts	100%	27.1%	42.9%	50.7%

network can thus learn to use this information when picking relevant statements from the memory. We fixed the embedding dimension to 100, learning rate to 0.01 and margin to 0.1 and performed 10 epochs of training. The test accuracies are reported in Table 1.

We also implemented the weakly supervised MemNN described in [8], using 3 layers and tying the weights of each layer to the same input/output matrix pair. We incorporated a positional encoding matrix as described in the paper. We set all the parameter values as described in the paper. Note that the supervised MemNN performs poorly on QA3, since we only considered 2 relevant memories (k = 2). In contrast, the weakly supervised MemNN performs better than the supervised network on this particular task, because we used 3 layers of input/output embedding in the weak network, so it is able to learn to recognize upto three supporting memories.

Table 2: Performance on Wiki QA Dataset.

Wiki QA			
	Train Accuracy	Test Accuracy	Speedup
LSTM (w/ entire article)	84.2%	30.1%	-
WMemNN (Basic)	88.9%	44.4%	1x
WMemNN + POS Pruning	73.6%	46.5%	4x
WMemNN + WV Pruning	72.1%	45.7%	5x

5.2 Wiki QA Task

After testing our MemNN and WMemNN implementations on the toy bAbI tasks, we trained the models on the more complex Wiki QA dataset [6] to analyze their performance on real-world data. The Wiki QA dataset is much larger (100-1000 statements per article, 40000+ words) so training took longer. Due to constraints on training time, we restricted our training and test set to only questions with a one word answer (most of which were yes/no type answers), discarding the rest. Further, we attempted to reduce the training time as well as improve accuracy, by pruning the input article to remove irrelevant statements using POS and word vector based pruning as described in Section 4.3. The results are displayed in Table 2. We compare the WMemNN implementations with a baseline LSTM implementation which was fed the entire wikipedia article followed by the question, after which an answer was sampled from it. The LSTM overfits to the sparse training set but is unable to generalize at all to test examples. A vanilla LSTM is ill-suited for this task because an entire Wikipedia article is too complex and noisy for an LSTM to learn any mapping from questions to answers. WMemNNs perform better than the baseline. Pruning gives a significant speedup in training time, and with a small change in test performance.

5.3 MCTest Task

Finally, we applied our WMemNN implementation to the MCTest dataset. The original paper [5] describes a baseline method using a combination of a sliding window score and a distance based score. We re-implemented the baseline method and could reproduce the accuracy numbers described in the paper. Next, we trained the WMemNN model on this task. Since this task consists of multiple choice answers instead of open ended answers, the MemNN models need to be tweaked accordingly. Specifically, the output in the MemNN (and WMemNN) is a sampled word sequence which is taken as the answer.

We tried two approaches to handle multiple choice questions: (i) First, we restricted our train/test datasets to questions which have only one word answers. The WMemNN model (also baseline LSTM) outputs a softmax probability vector over the entire vocabulary. The answer with the highest probability among the four choices is selected as the answer to the question. The results are shown in Table 3. We can observe that the LSTM baseline performs better than the WMemNN model. (ii) In our second approach, we extended the WMemNN as described in Section 4.4, which outputs a ranking over the four answer choices, so we take the one with the highest score. Unfortunately, our WMemNN implementations could not beat this simple baseline. We can see that the extended WMemNN model overfits the training set almost entirely. We tried the basic L2 regularization which did not help. Trying alternative regularization schemes should salvage this issue, but unfortunately we didn't have time to try this out. (See Section 6.1: Future Work.) We will attempt an error analysis in the next section.

Table 3: Performance on MCTest dataset, using the two approaches described in Section 5.3

	Approach 1: Restrict dataset		Approach 2: Incorporate choices into model			
Dataset	LSTM Test Accuracy	WMemNN (Basic) Test Accuracy	Baseline (SW + D)		WMemNN (Extended)	
			Train	Test	Train	Test
MC160	51.6%	45.2%	72.5%	66.2%	92.9%	36%
MC500	40.1%	36.5%	59.3%	56.7%	98.3%	34.2%

5.4 Error Analysis

Our MemNN and WMemNN implementations outperformed LSTMs on the first 3 bAbI tasks, which justifies the power of memory augmented models for large-scale QA tasks in which a simple LSTM model cannot remember everything from a long piece of text to correctly answer questions on it. However, the WMemNN performance on the real-world datasets was not satisfactory. In the Wiki QA task, the WMemNN model performs better than the baseline LSTM, but the performance is still less than that on the bAbI tasks. In the MCTest task, the WMemNN performs worse than the baseline. On manually inspecting the error cases, we found these general causes of errors:

1. **Anaphora resolution:** The most common source of error is when the statement containing the answer contain pronoun references to a noun mentioned in the previous sentences. Resolving such pronouns in the text before passing it into the WMemNN should lead to improvements.
2. **Identifying synonyms and hypernyms:** If the question paraphrases a piece of the input text or uses synonyms/hypernyms of words appearing in the text, the WMemNN model cannot handle this, and simple noun/NER based pruning is counterproductive in this case. For instance, for the text "John plays baseball.", the question could be "Which sport does John play?" One solution is to leverage pretrained word vectors (eg. word2vec [4]) to identify synonyms/hypernyms in the text and question and perform smarter pruning.
3. **Parsing complex sentence structures:** Another major problem is in parsing complex sentences. For example, for the text "Steve also liked bananas, oranges and apples, but fish was his favorite." and the question "What was Steve's favorite food?", the model selects "apples" instead of "fish". It is nearly impossible for the W matrix in the final output layer of WMemNN to learn the contrastive conjunction effects in sentences of the form "X but Y". Possibly, adding a recursive neural network component with a tensor product (RNTN) as done in Socher et al [7] could lead to better handling of such complex sentence structures.

We provide a detailed error analysis with sample failure cases for the MCTest task in Appendix A.

6 Conclusion

We studied the performance of memory augmented deep neural network based QA systems on one simulated dataset and two real-world datasets. Although the use of memory networks for large-scale QA tasks is clearly justified based on their performance on a simulated toy dataset, getting the memory network to work on more complex datasets requires substantial task-specific feature engineering

and model extensions. Overall, a hybrid two-pass approach of first effectively pruning the input statements using traditional NLP feature engineering, and then passing the pruned statements and question through a memory network to obtain the answer, seems to work best for complex QA tasks.

6.1 Future Work

Due to time constraints we left a few research directions unexplored in this work. For instance, in the Wiki QA task, we can tag named entities in the Wikipedia text and the question, and only keep statements in the article which contain or appear close to the NER entities present in the question. This should be a cheap way of speeding up training and prediction, as well as potentially improving accuracy, since we found that most questions in this task concerned only one or two named entities in the text.

The work on weakly supervised MemNNs in [8] does not touch upon the generative aspect of multiword answers. For any real world QA, or for an agent to maintain a dialog, the system must generate meaningful sentences. One way to accomplish this is to add an LSTM that is pre-trained on a language model, to the final layer of WMemNN to output sensible sentences in the answer.

Similarly, for the MCTest task, we observed that the simple heuristic baseline performed much better than the extended WMemNN implementation. Incorporate the baseline heuristic into the WMemNN input should increase accuracy overall. Secondly, the WMemNN model highly overfits the training set for this task, so it should be interesting to study the effects of adding regularization to the model cost function.

References

[1] Burges, Christopher JC. "Towards the Machine Comprehension of Text: An Essay". Microsoft Research Technical Report MSR-TR-2013-125, 2013, pdf, 2013.

[2] Fader, Anthony, Luke S. Zettlemoyer, and Oren Etzioni. "Paraphrase-Driven Learning for Open Question Answering." ACL (1). 2013.

[3] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural Turing Machines." arXiv preprint arXiv:1410.5401 (2014).

[4] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.

[5] Richardson, Matthew, et al. "MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text" EMNLP (2013)

[6] Smith, Noah A., et al. "Question Generation as a Competitive Undergraduate Course Project" In Proceedings of the NSF Workshop on the Question Generation Shared Task and Evaluation Challenge, Arlington, VA, 2008.

[7] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the conference on empirical methods in natural language processing (EMNLP). Vol. 1631. 2013.

[8] Sukhbaatar, Sainbayar, et al. "Weakly Supervised Memory Networks" arXiv preprint arXiv:1503.08895 (2015).

[9] Weston, Jason, et al. "Towards AI-complete question answering: A set of prerequisite toy tasks." arXiv preprint arXiv:1502.05698 (2015).

[10] Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

Appendix A MCTest Error Analysis

We provide a few failure cases of the WMemNN (extended) model on the MC160 QA task. For each example, we provide only the part of the input text that is relevant to the question. Each question has 4 choices. The correct answer is colored in green, and the choice guessed by the model is colored in red. Comments on that specific error are given in the caption.

Story: mc160.train.31

Ryan and Adam love to play basketball. They like it better than soccer and baseball. Their other friend, Jared, has his own basketball hoop. He got it for his ninth birthday. Ryan got a football for his birthday and Adam got a skateboard. They like their presents, but think the basketball hoop is better. They play basketball at Jared's house with him and any other kids who show up. Alex and Brady come almost every day and Josh, Ty, and Max come sometimes. Next year, they all get to play on a basketball team. They get to play at their school. They are very excited about that and can't wait to play on a real team. For now, they are practicing a lot and are trying to get really good. They play every day they can. They are trying to be as good as the NBA players they watch on TV. They dream of someday playing in the NBA. They are sure it is going to happen.

4: multiple: What did Jared get for his ninth birthday?
A) a basketball hoop B) a baseball bat C) a football D) a skateboard

Figure A.1: Here, the answer is split between two sentences, and the second sentence contains a pronoun reference for Jared, so the model is unable to pick this out. However, it matches the phrase "Ryan got a football for his birthday" to the question phrase "get for his ninth birthday" and subsequently selects "a football" as the answer.

Story: mc160.train.35

Once upon a time, there was a boy named Freddy. And Freddy loved his mom very much, and his mom loved him very much too. One day, Freddy went outside to ride his bike. On the way out, his mother told him, "Remember to wear your helmet," and Freddy grabbed his helmet and met his friends outside. When he was putting on his helmet, his friends told him, "Helmets are for girls! You're not cool if you wear a helmet!" Freddy thought about what his mom told him, but he wanted to be cool like his friends, and he took off his helmet.

1: one: Who does Freddy love?
A) His friends B) His bicycle C) His mom D) His dad

Figure A.2: This is actually an easy question which the model gets wrong. All three key words "Freddy, mom and love" appear in a single sentence of the input, but the WMemNN model is not able to identify it as the correct answer.

Story: mc160.train.36

There was a big race in town. Stephanie and Sarah were friends. Stephanie was faster than Sarah. On the day of the race, they wished each other good luck. Sarah tripped on a rock during the race. She cried but another one of her friends, Matt, helped her stand up. Stephanie cheered for her to finish after she crossed the line.

1: multiple: Based on the story, who likely won the race?
A) Jane B) Matt C) Sarah D) Stephanie

Figure A.3: This is hard for the model to get right since the question "who won the race?" is a paraphrase of the line "after [Stephanie] crossed the line." The model sees "Sarah" and "race" in the same sentence and so outputs Sarah as the answer.

Story: mc160.train.43

Bailey and her friend Kara were bored one Saturday. It was a hot summer day. They didn't want to stay inside any longer but they didn't know what to do. They were tired of watching TV inside. Suddenly, Kara had an idea. She said, "Bailey, we could make some money." "How?," asked Bailey. "Well, it is hot outside," said Kara. "People are thirsty out there. We could make money by making some lemonade and iced tea and have people pay for it." "That is a great idea," answered Bailey, "let's do it!" Kara had made some iced tea with her mom earlier that day. She asked her mom permission to use it. Her mom said yes. She and Kara made two pitchers of lemonade. They got a cooler full of ice and made a sign so people knew what was for sale. Kara's mom helped them get a table and chairs and set up out on the corner in their neighborhood. It was so hot out that people who saw their stand came to buy drinks right away. Their first visitors to their stand were their friends, Abby and Molly. In a half hour, they had to close their stand. They were all out of lemonade and iced tea. They had made a lot of money. They split the money and each got ten dollars. It was a great day.

3: multiple: Who help them set up their stand?
A) Abby's mom B) Bailey's mom C) Molly's mom D) Kara's mom

Figure A.4: Again in this case, the question contains a paraphrase of the input sentence containing the answer. The model cannot identify that "set up their stand" is a paraphrase of "helped them get a table and chairs...". It sees "Abby" and "stand" in the same statement and so gives highest score to "Abby's mom".

Story: mc160.train.52

Sarah is a girl. Sarah has one brother. Sarah's brother's name is Timothy. Sarah has one sister. Sarah's sister's name is Annabelle. Their last name is MacGregor. One day Sarah went to the park with her brother Timothy. They swung on the swings for a short time. Then Annabelle came out and swung with them. They all sang some nice songs together. They all became very happy. Then Timothy's friend came. Timothy liked his friend very much. Timothy went off the swing and went away with his friend. Then Annabelle and Sarah felt very very sad. Happily then Annabelle and Sarah's friend came. Their friend's name was Kate Smith. She was the same age as Sarah. They wanted to go to the slide together. So they went to the slide and played for a long time. Then Annabelle became happy. And Sarah also became happy. Then they went home together and had some food.

2: multiple: How many brothers and sisters does Sarah have?
A) 1 B) 2 C) 0 D) 3

Figure A.5: This is extremely difficult for the WMemNN model since it must perform numerical addition, as "Sarah has one brother" and "Sarah has one sister" appear separately in the input text.

Story: mc160.train.65

Grace wants to play Frisbee. She goes to her store to buy a Frisbee. She picks out a red Frisbee. It is small enough to fit in her hand. It costs 75 cents. She buys it. She leaves the store. When Grace gets home, she has no one to play with. She looks for her friend Susan. Susan is not at home. She looks for her friend Jeff. Jeff is not allowed to go outside. Grace finds a dog named Ginger. Ginger loves to play frisbee. Grace tosses the frisbee to Ginger. Ginger catches it in her mouth. Ginger brings the frisbee back to Grace. Grace tosses the frisbee again. Ginger jumps up in the air and catches it. Grace throws the Frisbee one more time. The Frisbee lands in a tree. Grace is too short to reach the Frisbee. Grace pets Ginger and tells her that she is a good girl. Grace takes Ginger home. They eat cookies. The next day, they come back to the park. They get their Frisbee back. They play again.

2: multiple: Who did Grace look for when she left the store?
A) Susan and Jeff B) Jeff and Grace C) Susan and Ginger D) Ginger and Jeff

Figure A.6: Again, this is difficult for the WMemNN model to get right, since it cannot identify that Ginger is a dog while Susan and Jeff are Grace's friends.