Long-Term Memory Networks for Question Answering

Fenglong Ma*1, Radha Chitta*2, Saurabh Kataria*3, Jing Zhou*2, Palghat Ramesh4, Tong Sun*5, Jing Gao1
1SUNY Buffalo, 2Conduent Labs US, 3LinkedIn
4PARC, 5United Technologies Research Center
{fenglong, jing}@buffalo.edu, {radha.chitta, jing.zhou}@conduent.edu
saurabh.cse05@gmail.com, palghat.ramesh@parc.com, sunt@utrc.utc.com

Abstract

Question answering is an important and difficult task in the natural language processing domain, because many basic natural language processing tasks can be cast into a question answering task. Several deep neural network architectures have been developed recently, which employ memory and inference components to memorize and reason over text information, and generate answers to questions. However, a major drawback of many such models is that they are capable of only generating single-word answers. In addition, they require large amount of training data to generate accurate answers. In this paper, we introduce the Long-Term Memory Network (LTMN), which incorporates both an external memory module and a Long Short-Term Memory (LSTM) module to comprehend the input data and generate multi-word answers. The LTMN model can be trained end-to-end using back-propagation and requires minimal supervision. We test our model on two synthetic data sets (based on Facebook’s bAbI data set) and the real-world Stanford question answering data set, and show that it can achieve state-of-the-art performance.

1 Introduction

Question answering (QA), a challenging problem which requires an ability to understand and analyze the given unstructured text, is one of the core tasks in natural language understanding and processing. Many problems in natural language processing, such as reading comprehension, machine translation, entity recognition, sentiment analysis, and dialogue generation, can be cast as question answering problems.

Traditional question answering approaches can be categorized as: (i) IR-based question answering [Pasca, 2003] where the question is formulated as a search query, and a short text segment is found on the Web or similar corpus for the answer; (ii) Knowledge-based question answering [Green Jr et al., 1961; Berant et al., 2013], which aims to answer a natural language question by mapping it to a semantic query over a database.

The traditional approaches are simple query-based techniques. It is difficult to establish the relationships between the sentences in the input text, and derive a meaningful representation of the information within the text using these traditional question-answering systems.

Figure 1: Example of a question answering task.
ing developed at Xerox PARC. Jobs had negotiated a visit to see the Xerox Alto computer and its Smalltalk development tools in exchange for Apple stock options.” is a supporting fact and extract the relevant portion of the supporting fact to form the answer. In addition, the model should have the ability to memorize all the facts that have been presented to it until the current time, and deduce the answer.

The authors of [Weston et al., 2015] proposed a new class of learning models named Memory Networks (MemNN), which use a long-term memory component to store information and an inference component for reasoning. [Kumar et al., 2016] proposed the Dynamic Memory Network (DMN) for general question answering tasks, which processes input sentences and questions, forms episodic memories, and generates answers. These two approaches are strongly supervised, i.e., only the supporting facts (factoids) are fed to the model as inputs for training the model for each type of question. For example, when training the model with the question in the fourth line of Figure 1, they employed to generate the first word of the answer, which is then multiple passes through the memory, to obtain the factoids which are relevant to each question. These factoids are then employed to generate the first word of the answer, which is then input to an LSTM unit. The LSTM unit is used to generate the subsequent words in the answer. The proposed LTMN model can be trained end-to-end, requires minimal supervision during training (i.e., weakly supervised), and generates multiple word answers. Experimental results on two synthetic datasets and one real world dataset show that the proposed model outperforms the state-of-the-art approaches.

In summary, the contributions of this paper are as follows:

- We propose an effective neural network architecture for general question answering, i.e., for generating multi-word answers for questions. Our architecture combines the best aspects of MemN2N and LSTM and can be trained end-to-end.
- The proposed architecture employs distributed representation learning techniques (e.g., paragraph2vec) to learn vector representations for sentences or factoids, questions and words, as well as their relationships. The learned embeddings contribute to the accuracy of the answers generated by the proposed architecture.
- We generate a new synthetic dataset with multiple word answers based on Facebook’s bAbI dataset [Weston et al., 2016]. We call this the multi-word answer bAbI dataset.
- We test the proposed architecture on two synthetic datasets (the single-word answer bAbI dataset and the multi-word answer bAbI dataset), and the real-world Stanford question answering dataset [Rajpurkar et al., 2016]. The results clearly demonstrate the advantages of the proposed architecture for question answering.

2 Related Work

In this section, we review literature closely related to question answering, particularly focusing on models using memory networks to generate answers.

2.1 Question Answering

Traditional question answering approaches mainly include two categories: IR-based [Pašca, 2003] and Knowledge-based question answering [Green Jr et al., 1961; Berant et al., 2013]. IR-based question answering systems use information retrieval techniques to extract information (i.e., answers) from documents. These methods first process questions, i.e., detect named entities in questions, and then predict answer types, such as cities’ names or person’s names. After recognizing answer types, these approaches generate queries, and extract answers from the web using the generated queries. These approaches are easy, but they ignore the semantics between questions and answers.

Knowledge-based question answering systems [Zettlemoyer and Collins, 2005; Berant and Liang, 2014; Zhang et al., 2016] consider the semantics and use existing knowledge bases, such as Freebase [Bollacker et al., 2008] and DBpedia [Bizer et al., 2009]. They cast the question answering task as that of finding one of the missing arguments in a triple. Most of knowledge-based question answering approaches use neural networks, dependency
trees and knowledge bases [Bordes et al., 2012] or sentences [Iyyer et al., 2014].

Using traditional question answering approaches, it is difficult to establish the relationship between sentences in the input text, and thereby identify the relevance of the different sentences to the question. Of late, several neural network architectures with memories have been proposed to solve this challenging problem.

2.2 Memory Networks

Several deep neural network models use memory architectures [Sukhbaatar et al., 2015; Kumar et al., 2016; Weston et al., 2015] to mimic human memory. The question module, like the input module, aims to encode the question representation and the sentence representations, and then outputs the weighted sum of the sentence representations and matching probabilities. Using this weighted sum vector and the question representation, the answer module finally generates the answer for the question.

3 Long-Term Memory Networks

In this section, we describe the proposed Long-Term Memory Network, shown in Figure 2. It includes four modules: input module, question module, memory module and answer module. The input module encodes raw text data (i.e., sentences) into vector representations. Similarly, the question module also encodes questions into vector representations. The input and question modules can use the same or different encoding methods. Given the input sentences’ representations, the memory module calculates the matching probabilities between the question representation and the sentence representations, and then outputs the weighted sum of the sentence representations and matching probabilities. Using this weighted sum vector and the question representation, the answer module finally generates the answer for the question.

3.1 Input Module and Question Module

Let \( \{x_i\}_{i=1}^n \) represent the set of input sentences. Each sentence \( x_i \in \mathbb{R}^{|V|} \) contains words belonging to a dictionary \( V \), and ends with an end-of-sentence token <EOS>. The goal of the input module is to encode sentences into vector representations. The question module, like the input module, aims to encode each question \( q \in \mathbb{R}^{|V|} \) into a vector representation. Specifically, we use a matrix \( A \in \mathbb{R}^{d \times |V|} \) to embed sentences and \( B \in \mathbb{R}^{d \times |V|} \) for questions.

Several methods have been proposed to encode the input sentences or questions. In [Sukhbaatar et al., 2015], an embedding matrix is employed to embed the sentences in a continuous space and obtain the vector representations. [Kumar et al., 2016; Elman, 1991] use a recurrent neural network to encode the input sentences into vector representations. Our objective is to learn the co-occurrence and sequence relationships between words in the text in order to generate a coherent sequence of words as answers. Thus, we employ a distributed representation learning technique, such as paragraph vectors (paragraph2vec) model [Le and Mikolov, 2014] to pre-train \( A \) and \( B \) (with \( A = B \)) for the real-word SQUAD dataset, which takes into account the order and semantics among words to encode the input sentences and questions.

For synthetic datasets, which are based on a small vocabulary, the embedding matrices \( A \) and \( B \) are learnt via back-propagation.

3.2 Memory Module

The input sentences \( \{x_i\}_{i=1}^n \) are embedded using the matrix \( A \) as \( m_i = Ax_i, i = 1, 2, \ldots, n; m_i \in \mathbb{R}^d \) and stored in memory. Note that we use all the sentences before the question as input, which implies that the proposed model is weakly supervised. The question \( q \) is also embedded using the matrix \( B \) as \( u = Bq; u \in \mathbb{R}^d \). The memory module then calculates the matching probabilities between the sentences and

\[ P(x_i | q) = \frac{e^{A u_m^i}}{\sum_{j=1}^n e^{A u_m^j}} \]

It can be trained end-to-end, (ii) it is weakly supervised, and (iii) can generate answers with multiple words.

---

1 We use paragraph2vec in our implementation. Other representation learning mechanisms may be employed in the proposed LTMN model.
the question, by computing the inner product followed by a softmax function as follows:
\[ p_i = \text{softmax}(u^T m_i), \]  
(1)
where \( \text{softmax}(z_i) = e^{z_i} / \sum_j e^{z_j} \). The probability \( p_i \) is expected to be high for all the sentences \( x_i \) that are related to the question \( q \).

The output of the memory module is a vector \( o \in \mathbb{R}^d \), which can be represented by the sum over input sentence representations, weighted by the matching probability vector as follows:
\[ o = \sum_i p_i m_i. \]  
(2)

This approach, known as the **soft attention mechanism**, has been used by [Sukhbaatar et al., 2015; Bahdanau et al., 2015]. The benefit of this approach is that it is easy to compute gradients and back-propagate through this function.

### 3.3 Answer Module

Based on the output vector \( o \) from the memory module and the word representations from input module, the answer module generates answers for questions. As our objective is to generate answers with **multiple words**, we employ Long Short Term Memory Networks (LSTM) [Hochreiter and Schmidhuber, 1997] to generate answers.

The core of the LSTM neural network is a memory unit whose behavior is controlled by a set of three gates: input, forget, and output gates. The memory unit accumulates the knowledge from the input data at each time step, based on the values of the gates, and stores this knowledge in its internal state. The initial input to the LSTM is the embedding of the token and its state. We use the output of the memory module \( o \), the question representation \( u \), a weight matrix \( W^{(o)} \) and bias \( b_o \) to generate the embedding of \( \text{<BOA>} \) \( a_0 \) as follows:
\[ a_0 = \text{softmax}(W^{(o)}(o + u) + b_o). \]  
(3)

Using \( a_0 \) and the initial state \( s_0 \), LSTM can generate the first word \( w_1 \) and its corresponding predicted output \( y_1 \) and state \( s_1 \). At each time step \( t \), LSTM takes the embedding of word \( w_{t-1} \) and last hidden state \( s_{t-1} \) as input to generate the new word \( w_t \).
\[ v_t = [w_{t-1}] \]  
(4)

\[ i_t = \sigma(W_{iv}v_t + W_{im}y_{t-1} + b_i) \]  
(5)
\[ f_t = \sigma(W_{fv}v_t + W_{fm}y_{t-1} + b_f) \]  
(6)
\[ o_t = \sigma(W_{ov}v_t + W_{om}y_{t-1} + b_o) \]  
(7)
\[ s_t = i_t \odot s_{t-1} + f_t \odot \tanh(W_{sv}v_t + W_{sm}y_{t-1}) \]  
(8)
\[ y_t = o_t \odot s_t \]  
(9)
\[ w_t = \text{argmax} \left[ \text{softmax}(W^{(t)}y_t + b_t) \right] \]  
(10)

where \([w_t]\) is the embedding of word \( w_t \) learnt from the input module, \( \sigma \) and \( \odot \) denote the sigmoid function and Hadamard product respectively, and \( W^{(t)} \) is a weight matrix and \( b_t \) is a bias vector.

The model is trained end-to-end with the loss defined by the cross-entropy between the true answer and the predicted output \( w_t \), represented using one-hot encoding. The predicted answer is generated by concatenating all the words generated by the model.

### 4 Experiments

In this section, we compare the performance of the proposed LTNN model with the current state-of-the-art models for question answering.

#### 4.1 Datasets

We use three datasets: the real-world Stanford question answering dataset (SQuAD) [Rajpurkar et al., 2016], the synthetic single-word answer bAbI dataset [Weston et al., 2016], and the synthetic multi-word answer bAbI dataset, generated by performing vocabulary replacements in the single-word answer bAbI dataset.

**Stanford Question Answering Dataset (SQuAD)** [Rajpurkar et al., 2016] contains 100,000+ questions labeled by crowd workers on a set of Wikipedia articles. The answer for each question is a segment of text from the corresponding paragraph. In order to convert the format of the data to the input format of our model (shown in Figure 1), we use NLTK to detect the boundary of sentences and assign an index to each sentence and question, in accordance with the starting index of the answer provided by the crowd workers. The dataset is thus transformed to a question answer dataset containing 18,893 stories and
69,523 question. For our experiments, we randomly selected 1,248 questions for training and 1,248 questions for testing. Each answer contains less than or equal to five words.

The single-word answer bAbI dataset [Weston et al., 2016] is a synthetic dataset created to benchmark question answering models. It contains 20 types of question answer tasks, and each task is comprising a set of statements followed by a single-word answer. For each question, only some of the statements contain the relevant information. The training and test data contains 1,000 examples for each task.

The multi-word answer bAbI dataset. As the goal of the proposed model is to generate multi-word answers, we manually generated a new dataset from the Facebook bAbI dataset, by replacing few words, such as "bedroom" and "bathroom" with "guest room", and "shower room", respectively. The replacements are listed in Table 1.

Table 1: Replacements made in the vocabulary of the bAbI dataset to generate the multi-word answer bAbI dataset.

| Original word       | Replacement          |
|---------------------|----------------------|
| hallway             | entrance way         |
| bathroom            | shower room          |
| office              | computer science office |
| bedroom             | guest room           |
| milk                | hot water            |
| Bill                | Bill Gates           |
| Fred                | Fred Bush            |
| Mary                | Mary Bush            |
| green               | bright green         |
| yellow              | bright yellow        |
| hungry              | extremely hungry     |
| tired               | extremely tired      |

4.2 Parameters and Baselines

We use 10% of the training data for model validation to choose the best parameters. The best performance was obtained when the learning rate was set to 0.002, the batch size set to 32, and the weights initialized randomly from a Gaussian distribution with zero mean and 0.1 variance. The model was trained for 200 epochs. The paragraph2vec model was set to generate 100-dimensional representations for the input sentences and the questions.

We first compare the performance of the proposed LTMN model with a simple Long Short Term Memory network (LSTM) model, as implemented in [Sutskever et al., 2014] to predict sequences. The LSTM model works by reading the story until it comes across a question and outputs an answer, using the information obtained from the sentences read so far. Unlike the LTMN model, it does not have an external memory component. We also compare its performance on the single-word answer bAbI dataset, we also compare our results with those of the at-
tention based LSTM model (LSTM + Attention) [Hermann et al., 2015], which propagates dependencies between input sentences using an attention mechanism. MemNN [Weston et al., 2015], DMN [Kumar et al., 2016], and MemN2N [Sukhbaatar et al., 2015]. These models cannot be applied as-is to the SQuAD and multi-word answer bAbI datasets because they are only capable of generating single-word answers.

4.3 Evaluation Measures

In order to evaluate the performance of all the methods, the following measurements are used:

- Exact Match Accuracy (EMA) represents the ratio of predicted answers which exactly match the true answers.
- Partial Match Accuracy (PMA) is the ratio of generated answers that partially match the correct answers.
- BLEU score [Chen and Cherry, 2014], widely used to evaluate machine translation models, measures the quality of the generated answers.

Table 2: Test accuracy on the SQuAD dataset.

| Measure | LSTM | LTMN |
|---------|------|------|
| EMA     | 8.3  | 10.6 |
| BLEU    | 12.4 | 17.0 |
| PMA     | 22.8 | 27.4 |

4.4 Results

The performance of the LTMN model is shown in Tables 2, 3, and 4 on the SQuAD, single-word answer bAbI and multi-word answer bAbI datasets, respectively.

We observe that LTMN performs better than LSTM in terms of all three evaluation measures, on all the datasets. On the SQuAD dataset, as the vocabulary is large (8,969), the LSTM model cannot learn the embedding matrices accurately, leading to its poor performance. However, as the LTMN model employs paragraph2vec, it learns richer vector representations of the sentences and questions. In addition, it can memorize and reason over the facts better than the simple LSTM model. On the multi-word answer bAbI dataset, the LTMN model is significantly better than the LSTM model, especially on tasks 1, 4, 12, 15, 19, and 20. The average EMA, BLEU, and PMA scores of LTMN are about 30% higher than those of the LSTM model. The single-word answer bAbI dataset’s vocabulary is small (about 20), so we learn the embedding matrices A and B using back-propagation, instead of using paragraph2vec to obtain the vector representations. In Table 3 we observe that the LTMN model achieves accuracy close to the strongly supervised MemNN and DMN models on 4 out of the 20 bAbI tasks, despite being weakly supervised, and achieves better accuracy than the weakly-supervised LSTM+Attention and MemN2N on 7 tasks. The proposed LTMN model also offers the additional capability of generating multi-word answers, unlike these baseline models.

The dataset can be downloaded from http://www.acsu.buffalo.edu/~fenglong/
Table 3: Test accuracy (EMA) on the single-word answer bAbI dataset

| Task                           | Weakly Supervised | Strongly Supervised |
|-------------------------------|-------------------|---------------------|
|                               | LSTM              | LSTM + Attention    | MemN2N | LTWN   | MemNN | DMN   |
| 1: Single Supporting Fact     | 50                | 98.1                | 96     | 98.2   | 100   | 100   |
| 2: Two Supporting Facts       | 20                | 33.6                | 61     | 41.6   | 100   | 98.2  |
| 3: Three Supporting Facts     | 20                | 25.5                | 30     | 23.8   | 100   | 95.2  |
| 4: Two Argument Relations     | 61                | 98.5                | 93     | 98.1   | 100   | 100   |
| 5: Three Argument Relations   | 70                | 97.8                | 81     | 79.5   | 98    | 99.3  |
| 6: Yes/No Questions           | 48                | 55.6                | 72     | **81.8** | 100   | 100   |
| 7: Counting                   | 49                | 80.0                | 80     | **80.2** | 85    | 96.9  |
| 8: Lists/Sets                 | 45                | 92.1                | 77     | 72.6   | 91    | 96.5  |
| 9: Simple Negation            | 64                | 64.3                | 72     | 65.4   | 100   | 100   |
| 10: Indefinite Knowledge      | 46                | 57.2                | 63     | **87.0** | 98    | 97.5  |
| 11: Basic Coreference         | 62                | 94.4                | 89     | 84.7   | 100   | 99.9  |
| 12: Conjunction               | 74                | 93.6                | 92     | **97.9** | 100   | 100   |
| 13: Compound Coreference      | 94                | 94.4                | 93     | 90.3   | 100   | 99.8  |
| 14: Time Reasoning            | 27                | 75.3                | 76     | 74.3   | 99    | 100   |
| 15: Basic Deduction           | 21                | 57.6                | 100    | **100** | 100   | 100   |
| 16: Basic Induction           | 23                | 50.4                | 46     | 43.5   | 100   | 99.4  |
| 17: Positional Reasoning      | 51                | 63.1                | 57     | 57.0   | 65    | 59.6  |
| 18: Size Reasoning            | 52                | 92.7                | 90     | 90.7   | 95    | 95.3  |
| 19: Path Finding              | 8                 | 11.5                | 9      | 11.4   | 36    | 34.5  |
| 20: Agent’s Motivations       | 91                | 98.0                | 100    | **100** | 100   | 100   |
| Mean (%)                      | 48.8              | 71.7                | 73.9   | 73.9   | 93.4  | 93.6  |

Table 4: Test accuracy on the multi-word answer bAbI dataset.

| Task                           | LSTM | LSTM + Attention | MemN2N | LTWN   | MemNN | DMN   |
|-------------------------------|------|------------------|--------|--------|-------|-------|
|                               | EMA  | BLEU             | PMA    | EMA    | BLEU  | PMA   |
| 1: Single Supporting Fact     | 36.5 | 38.8             | 41.1   | **97.0** | **97.2** | **97.3** |
| 2: Two Supporting Facts       | 26.6 | 29.7             | 32.7   | 31.3   | 34.5  | 37.6  |
| 3: Three Supporting Facts     | 17.1 | 20.3             | 23.6   | 24.5   | 27.2  | 29.8  |
| 4: Two Argument Relations     | 48.2 | 50.1             | 51.9   | **97.9** | **98.0** | **98.0** |
| 5: Three Argument Relations   | 45.3 | 49.3             | 53.2   | 77.9   | 80.1  | 82.2  |
| 6: Yes/No Questions           | 53.8 | 53.8             | 53.8   | 66.1   | 66.1  | 66.1  |
| 7: Counting                   | 69.5 | 69.5             | 69.5   | 78.4   | 78.4  | 78.4  |
| 8: Lists/Sets                 | 62.1 | 66.7             | 71.8   | 82.1   | 85.6  | 89.3  |
| 9: Simple Negation            | 57.4 | 57.4             | 57.4   | 69.2   | 69.2  | 69.2  |
| 10: Indefinite Knowledge      | 44.4 | 44.4             | 44.4   | 84.7   | 84.7  | 84.7  |
| 11: Basic Coreference         | 33.1 | 35.1             | 37.0   | 83.3   | 83.7  | 84.0  |
| 12: Conjunction               | 33.1 | 35.7             | 38.2   | 87.7   | 88.5  | 89.2  |
| 13: Compound Coreference      | 33.6 | 35.8             | 37.9   | 74.4   | 74.4  | 74.4  |
| 14: Time Reasoning            | 24.6 | 24.6             | 24.6   | 100    | 100   | 100   |
| 15: Basic Deduction           | 46.4 | 46.4             | 46.4   | **100** | **100** | **100** |
| 16: Basic Induction           | 46.8 | 51.6             | 56.3   | 42.4   | 47.0  | 51.6  |
| 17: Positional Reasoning      | 55.1 | 55.1             | 55.1   | 55.5   | 55.5  | 55.5  |
| 18: Size Reasoning            | 51.9 | 51.9             | 51.9   | 89.6   | 89.6  | 89.6  |
| 19: Path Finding              | 8.1  | 35.1             | 56.4   | 11.3   | 59.1  | **100** |
| 20: Agent’s Motivations       | 83.3 | 84.6             | 85.3   | **100** | **100** | **100** |
| Mean (%)                      | 42.2 | 46.8             | 49.4   | 72.6   | 75.9  | 78.8  |

5 Conclusions

Question answering is an important and challenging task in natural language processing. Traditional question answering approaches are simple query-based approaches, which cannot memorize and reason over the input text. Deep neural networks with memory have been employed to alleviate this challenge in the literature.

In this paper, we proposed the Long-Term Memory Network, a novel recurrent neural network, which can encode raw text information (the input sentences and questions) into vector representations, form memories, find relevant information in the input sentences to answer the questions, and finally generate multi-word answers using a long short term memory...
network. The proposed architecture is a weakly supervised model and can be trained end-to-end. Experiments on both synthetic and real-world datasets demonstrate the remarkable performance of the proposed architecture.

In our experiments on the bAbI question & answering tasks, we found that the proposed model fails to perform as well as the completely supervised memory networks on certain tasks. In addition, the model performs poorly when the input sentences are very long and the vocabulary is large, as it cannot calculate the supporting facts efficiently. In the future, we plan to expand the model to handle long input sentences, and improve the performance of the proposed network.

References
[Bahdanau et al., 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In ICLR, 2015.

[Berant and Liang, 2014] Jonathan Berant and Percy Liang. Semantic parsing via paraphrasing. In ACL, 2014.

[Berant et al., 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In EMNLP, 2013.

[Bizer et al., 2009] Christian Bizer, Jens Lehmann, Georgi Kiblarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. Web Semant, 2009.

[Bollacker et al., 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD, 2008.

[Bordes et al., 2012] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In AISTATS, 2012.

[Chen and Cherry, 2014] Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In SMT, 2014.

[Cho et al., 2014] 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.

[Elman, 1991] Jeffrey L Elman. Distributed representations, simple recurrent networks, and grammatical structure. Machine learning, 1991.

[Graves et al., 2014] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. arXiv preprint arXiv:1410.5401, 2014.

[Graves, 2013] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[Green Jr et al., 1961] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laugherly. Baseball: an automatic question-answerer. In Western joint IRE-AIEE-ACM computer conference, 1961.

[Hermann et al., 2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In NIPS, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 1997.

[Iyyer et al., 2014] Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In EMNLP, 2014.

[Joulin and Mikolov, 2015] Armand Joulin and Tomas Mikolov. Inferring algorithmic patterns with stack-augmented recurrent nets. In NIPS, 2015.

[Kumar et al., 2016] Ankit Kumar, Ozan Irsay, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In ICML, 2016.

[Le and Mikolov, 2014] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In ICML, 2014.

[Ma et al., 2017] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In KDD, 2017.

[Mozzer and Das, 1993] Michael C Mozer and Srerupa Das. A connectionist symbol manipulator that discovers the structure of context-free languages. In NIPS, 1993.

[Paśca, 2003] Marius Paśca. Open-domain question answering from large text collections. Computational Linguistics, 2003.

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

[Sukhbaatar et al., 2015] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In NIPS, 2015.

[Suo et al., 2017] Qiuling Suo, Fenglong Ma, Giovanni Canino, Jing Gao, Aidong Zhang, Pierangelo Veltri, and Agostino Gnamo. A multi-task framework for monitoring health conditions via attention-based recurrent neural networks. In AMIA, 2017.

[Sutskever et al., 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In NIPS, 2014.

[Wang et al., 2016] Bingning Wang, Shangmin Guo, Kang Liu, Shizhu He, and Jun Zhao. Employing external rich knowledge for machine comprehension. In IJCAI, 2016.

[Weston et al., 2015] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In ICLR, 2015.

[Weston et al., 2016] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In ICLR, 2016.

[You et al., 2016] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In CVPR, 2016.

[Zettlemoyer and Collins, 2005] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In UAI, 2005.

[Zhang et al., 2016] Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. A joint model for question answering over multiple knowledge bases. In AAAI, 2016.