ASK ME ANYTHING : DYNAMIC MEMORY NETWORKS

Akshat Jindal(150075) Soumye Singhal(150728) Abhisek Panda(150026)
Siddharth Mittal(150718) K. Siddarth(150312) Ginuga Saketh(150257)

April 13, 2018

- What is QA?
  Put simply,Question answering (QA) is a complex natural language processing task which requires an understanding of the meaning of a text and the ability to reason over relevant facts.

- What is QA?
  Put simply,Question answering (QA) is a complex natural language processing task which requires an understanding of the meaning of a text and the ability to reason over relevant facts.

- **What is QA?**

  Put simply,Question answering (QA) is a complex natural language processing task which requires an understanding of the meaning of a text and the ability to reason over relevant facts.

- **Why the hype?**

  Question answering as an NLP task can model various other tasks as well. Sequence modelling tasks like NER and POS tagging, classfiation tasks like sentiment analysis etc can be cast as QA problems.

- **What is QA?**
  Put simply,Question answering (QA) is a complex natural language processing task which requires an understanding of the meaning of a text and the ability to reason over relevant facts.
- **Why the hype?**
  Question answering as an NLP task can model various other tasks as well. Sequence modelling tasks like NER and POS tagging, classfiation tasks like sentiment analysis etc can be cast as QA problems.

INTRODUCTION

- Traditional Approaches

  Original QA systems often involved developing a structured knowledge database that is hand-written by experts in a specific domain. In these systems, a question asked in natural language must be parsed and converted into a machine-understandable query that returns the appropriate answer.

- **Traditional Approaches**

  Original QA systems often involved developing a structured knowledge database that is hand-written by experts in a specific domain. In these systems, a question asked in natural language must be parsed and converted into a machine-understandable query that returns the appropriate answer.

- **Traditional Approaches**

  Original QA systems often involved developing a structured knowledge database that is hand-written by experts in a specific domain. In these systems, a question asked in natural language must be parsed and converted into a machine-understandable query that returns the appropriate answer.

- **Current Approach**

  With advancements in deep learning, most QA systems nowadays use RNNs for the same. They generate latent representations of natural language text passages rather than relying on extracted features such as part of speech tagging,parsing, named entity recognition, etc. These networks have outperformed traditional models by huge margins.

- **Traditional Approaches**

  Original QA systems often involved developing a structured knowledge database that is hand-written by experts in a specific domain. In these systems, a question asked in natural language must be parsed and converted into a machine-understandable query that returns the appropriate answer.

- **Current Approach**

  With advancements in deep learning, most QA systems nowadays use RNNs for the same. They generate latent representations of natural language text passages rather than relying on extracted features such as part of speech tagging,parsing, named entity recognition, etc. These networks have outperformed traditional models by huge margins.

DYNAMIC MEMORY NETWORKS

DMNS

1. Our project is baselined on implementing (DMN), a neural network architecture which processes input sequences and questions, forms episodic memories, and generates relevant answers.

1. Our project is baselined on implementing (DMN), a neural network architecture which processes input sequences and questions, forms episodic memories, and generates relevant answers.
2. Questions trigger an iterative attention process which allows the model to condition its attention on the inputs and the result of previous iterations.

DMNS

1. Our project is baselined on implementing (DMN), a neural network architecture which processes input sequences and questions, forms episodic memories, and generates relevant answers.
2. Questions trigger an iterative attention process which allows the model to condition its attention on the inputs and the result of previous iterations.
3. These results are then reasoned over in a hierarchical recurrent sequence model to generate answers.

DMNS

1. Our project is baselined on implementing (DMN), a neural network architecture which processes input sequences and questions, forms episodic memories, and generates relevant answers.
2. Questions trigger an iterative attention process which allows the model to condition its attention on the inputs and the result of previous iterations.
3. These results are then reasoned over in a hierarchical recurrent sequence model to generate answers.
4. While the DMN can be trained end-to-end for a variety of tasks, we trained it for question answering on the Facebook baBi Dataset.

DATASET

DATASET

1. The Facebook bAbI-10k dataset consists of 20 tasks .Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.

DATASET

1. The Facebook bAbI-10k dataset consists of 20 tasks .Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.
2. Training Examples : 10,000   Test Examples : 1000

1. The Facebook bAbI-10k dataset consists of 20 tasks .Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.
2. **Training Examples** : 10,000 **Test Examples** : 1000
3. All examples consist of an input-question-answer tuple. The input is a variable length passage of text. The type of question and answer depends on the task.

DATASET

1. The Facebook bAbI-10k dataset consists of 20 tasks .Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.
2. **Training Examples** : 10,000 **Test Examples** : 1000
3. All examples consist of an input-question-answer tuple. The input is a variable length passage of text. The type of question and answer depends on the task.
4. For each question-answer pair, the dataset also gives the line numbers of the input passage that is relevant to the answer.

1. The Facebook bAbI-10k dataset consists of 20 tasks .Each task has a different type of question such as single supporting fact questions, two supporting fact questions, yes no questions, counting questions, etc.
2. **Training Examples** : 10,000 **Test Examples** : 1000
3. All examples consist of an input-question-answer tuple. The input is a variable length passage of text. The type of question and answer depends on the task.
4. For each question-answer pair, the dataset also gives the line numbers of the input passage that is relevant to the answer.
5. Every answer in the bAbI dataset is one word.

A PEEK INTO BABI

```
Two supporting fact example:

1 Mary got the milk there.
2 John moved to the bedroom.
3 Sandra went back to the kitchen.
4 Mary travelled to the hallway.
5 Where is the milk?   hallway   1 4
```

```
Yes/no question example:

2 John moved to the bedroom.
3 Is John in the kitchen?   no   2
```

DMN MODULES

The DMN model consists of 4 modules

1. Input module

MODEL DESCRIPTION

The DMN model consists of 4 modules

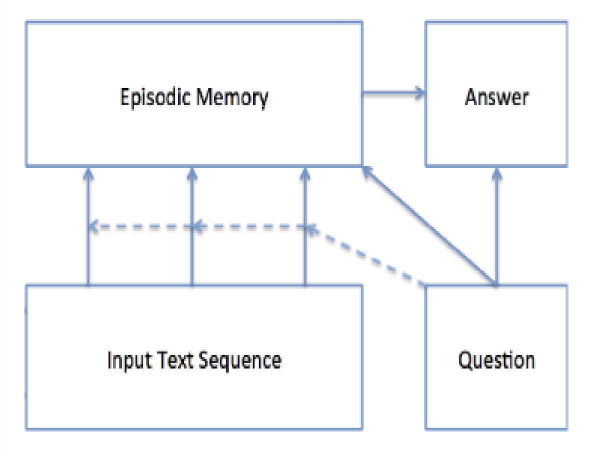
1. Input module
2. Question module

The DMN model consists of 4 modules

1. Input module
2. Question module
3. Episodic memory module

The DMN model consists of 4 modules

1. Input module
2. Question module
3. Episodic memory module
4. Answer module


Episodic Memory
Answer
Input Text Sequence
Question

1. The input module feeds word vectors through a GRU, and outputs the hidden states at the end of each sentence for the episodic memory module.

$$h_t = \mathrm{GRU}(L[w_t], h_{t-1})$$

INPUT, QUESTION AND ANSWER MODULES

1. The input module feeds word vectors through a GRU, and outputs the hidden states at the end of each sentence for the episodic memory module.

$$h_t = \mathrm{GRU}(L[w_t], h_{t-1})$$

2. The question module also takes word vectors as input, but outputs the question encoding.

1. The input module feeds word vectors through a GRU, and outputs the hidden states at the end of each sentence for the episodic memory module.

$$h_t = \mathrm{GRU}(L[w_t], h_{t-1})$$

2. The question module also takes word vectors as input, but outputs the question encoding.
3. The answer module uses GRU and at each time step takes as input the question q, the last hidden state a_{t-1} along with previous output. Finally, a softmax activation is applied. This produces a distribution over answer tokens.

$$a_t = \mathrm{GRU}([y_{t-1}, q], a_{t-1})$$
$$y_t = \mathrm{softmax}(W^{(a)} a_t)$$

INPUT, QUESTION AND ANSWER MODULES

1. The input module feeds word vectors through a GRU, and outputs the hidden states at the end of each sentence for the episodic memory module.

$$h_t = \mathrm{GRU}(L[w_t], h_{t-1})$$

2. The question module also takes word vectors as input, but outputs the question encoding.
3. The answer module uses GRU and at each time step takes as input the question q, the last hidden state a_{t-1} along with previous output. Finally, a softmax activation is applied. This produces a distribution over answer tokens.

$$a_t = \mathrm{GRU}([y_{t-1}, q], a_{t-1})$$
$$y_t = \mathrm{softmax}(W^{(a)} a_t)$$

4. The output is trained with cross entropy error classification of the correct sequence.

1. The episodic module reasons over the sentence states from input and question modules to produce a final state which the answer module uses as input.

EPISODIC MODULE

1. The episodic module reasons over the sentence states from input and question modules to produce a final state which the answer module uses as input.
2. The current sentence state c_t, the memory state m and question state q are used to determine if the current state should retained, or if we should look at the new sentence.

1. The episodic module reasons over the sentence states from input and question modules to produce a final state which the answer module uses as input.
2. The current sentence state c_t, the memory state m and question state q are used to determine if the current state should retained, or if we should look at the new sentence.
3. For each sentence, the attention mechanism updates states as-

$$z_t^i = [c_t, m^{i-1}, q, c_t \circ q, c_t \circ m^{i-1}, |c_t - q|, |c_t - m^{i-1}|]$$
$$g_t^i = G(c_t, m^{i-1}, q) = \sigma(W^{(2)}\tanh(W^{(1)}z_t^i + b^{(1)}) + b^{(2)})$$
$$h_t^i = g_t^i GRU(c_t, h_{t-1}^i) + (1 - g_t^i)h_{t-1}^i$$
$$e^i = h_{T_c}$$

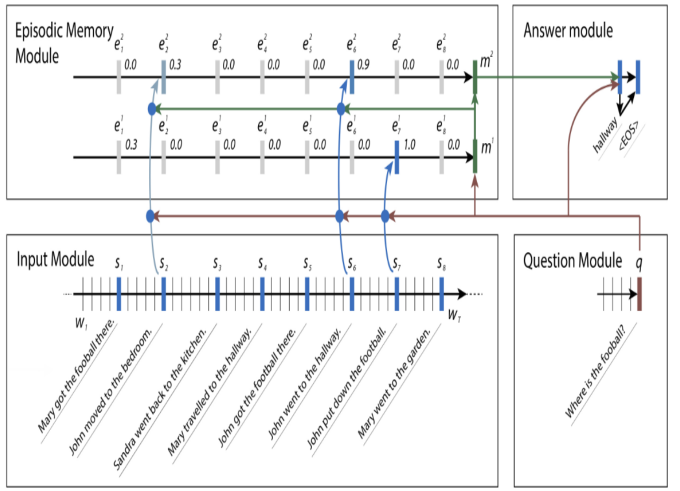
1. The episodic module reasons over the sentence states from input and question modules to produce a final state which the answer module uses as input.
2. The current sentence state c_t, the memory state m and question state q are used to determine if the current state should retained, or if we should look at the new sentence.
3. For each sentence, the attention mechanism updates states as-

$$z_t^i = [c_t, m^{i-1}, q, c_t \circ q, c_t \circ m^{i-1}, |c_t - q|, |c_t - m^{i-1}|]$$
$$g_t^i = G(c_t, m^{i-1}, q) = \sigma(W^{(2)} \tanh(W^{(1)} z_t^i + b^{(1)}) + b^{(2)})$$
$$h_t^i = g_t^i GRU(c_t, h_{t-1}^i) + (1 - g_t^i) h_{t-1}^i$$
$$e^i = h_{T_c}$$

4. The memory is then updated using the current episode state and the memory state by passing through a GRU.

$$m^i = GRU(e^i, m^{i-1})$$

NEED FOR MULTIPLE EPISODES



NEED FOR MULTIPLE EPISODES

1. In this example, for the question ”Where is the football?”. So, after the first pass, the attention mechanism gives more score to lines like ”Mary got the football there”, ”John put down the football”.

NEED FOR MULTIPLE EPISODES

1. In this example, for the question ”Where is the football?”. So, after the first pass, the attention mechanism gives more score to lines like ”Mary got the football there”, ”John put down the football”.
2. Since the memory is initialised with question embedding, the attention module retrieves all sentences containing ”football”.

NEED FOR MULTIPLE EPISODES

1. In this example, for the question ”Where is the football?”. So, after the first pass, the attention mechanism gives more score to lines like ”Mary got the football there”, ”John put down the football”.
2. Since the memory is initialised with question embedding, the attention module retrieves all sentences containing ”football”.
3. Then, by using this memory after the first pass, the attention mechanism the focuses on sentences like ”John went to the hallway”, since it infers a connection between ”football” and ”John”

NEED FOR MULTIPLE EPISODES

1. In this example, for the question ”Where is the football?”. So, after the first pass, the attention mechanism gives more score to lines like ”Mary got the football there”, ”John put down the football”.
2. Since the memory is initialised with question embedding, the attention module retrieves all sentences containing ”football”.
3. Then, by using this memory after the first pass, the attention mechanism the focuses on sentences like ”John went to the hallway”, since it infers a connection between ”football” and ”John”
4. Hence, multiple passes through input sequence helps the model to make transitive inferences.

OUR MODIFICATIONS

OUR MODIFICATIONS

1. Memory update mechanism:

OUR MODIFICATIONS

1. Memory update mechanism:
    - Using GRU conditioned on memory from previous timestep

$$m^i = \mathrm{GRU}(e^i, m^{i-1})$$

1. Memory update mechanism:
   - Using GRU conditioned on memory from previous timestep

$$m^i = \mathrm{GRU}(e^i, m^{i-1})$$

   - Using feedforward neural network over question, input facts and memory from previous iteration

$$m^i = \mathrm{PRelu}(W[m^{i-1}; Q; e^i] + b)$$

OUR MODIFICATIONS

1. Memory update mechanism:
    - Using GRU conditioned on memory from previous timestep

$$m^i = \text{GRU}(e^i, m^{i-1})$$

    - Using feedforward neural network over question, input facts and memory from previous iteration

$$m^i = \text{PRelu}(W[m^{i-1}; Q; e^i] + b)$$

2. Feature Vectors for Scoring Network:

OUR MODIFICATIONS

1. Memory update mechanism:
   - Using GRU conditioned on memory from previous timestep

$$m^i = GRU(e^i, m^{i-1})$$

   - Using feedforward neural network over question, input facts and memory from previous iteration

$$m^i = PRelu(W[m^{i-1}; Q; e^i] + b)$$

2. Feature Vectors for Scoring Network:
   - Custom: $[c_t \circ q, c_t \circ m, |c_t - q|, |c_t - m|]$

OUR MODIFICATIONS

1. Memory update mechanism:
    - Using GRU conditioned on memory from previous timestep

$$m^i = \mathrm{GRU}(e^i, m^{i-1})$$

    - Using feedforward neural network over question, input facts and memory from previous iteration

$$m^i = \mathrm{PRelu}(W[m^{i-1}; Q; e^i] + b)$$

2. Feature Vectors for Scoring Network:
    - Custom: $[c_t \circ q, c_t \circ m, |c_t - q|, |c_t - m|]$
    - Simple: $[c_t, m, q]$

EXPERIMENTAL RESULTS

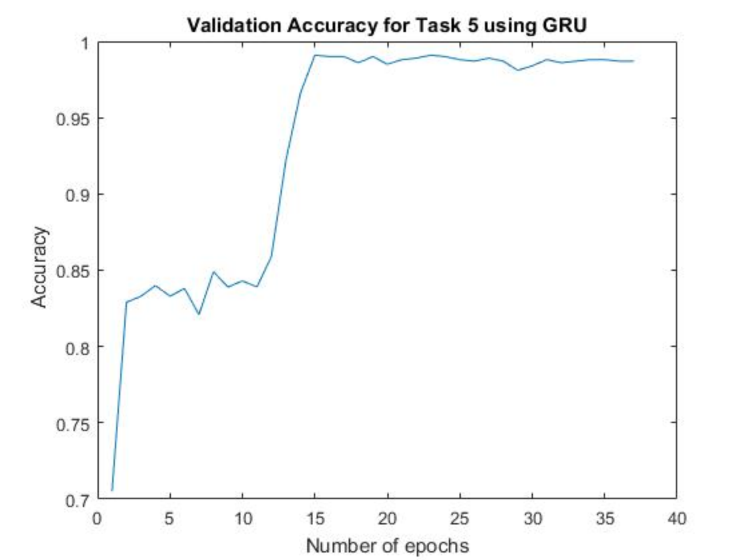
Table: Comparision of test accuracies of various architectures

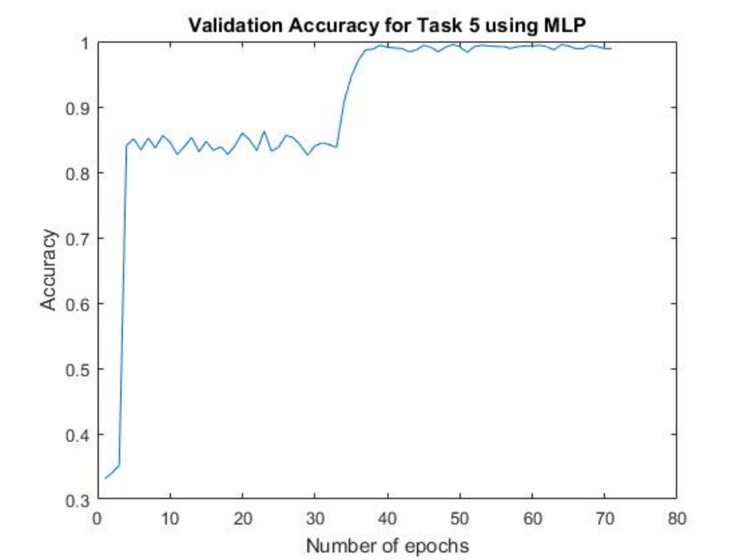
Task	DMN (GRU+custom)	DMN (FNN+custom)	DMN (GRU+simple)	DMN (FNN+simple)
1	100	100	100	100
2	78.5	87.7	88.2	89.6
3	44.8	54.9	43.7	42.2
4	100	100	100	100
5	94.6	98.9	98.3	98.8
6	50.5	49.6	49	49.7
7	97.6	97.1	96.6	95.8
8	99.4	99.2	99.5	99.3
9	99.0	90.7	100	79.8
10	88.4	51	52	50.3

Table: Comparision of test accuracies for various architectures(continued)

Task	DMN (GRU+custom)	DMN (FNN+custom)	DMN (GRU+simple)	DMN (FNN+simple)
11	94.6	93.0	100	84.8
12	100	100	100	100
13	91.5	91.7	91.8	93.7
14	64.1	32.5	18.1	47.3
15	100	98.2	99.3	100
16	44.2	44.4	44.1	45.2
17	59.1	62.8	60.4	62.2
18	91.1	91.2	90.1	90.2
19	35.1	45.8	41.2	31.2
20	99.2	100	96	90.9

ACCURACY CURVES




Validation Accuracy for Task 5 using MLP
Accuracy
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50
60
70
80
Number of epochs

POSSIBLE EXTENSIONS

1. Currently we are using GRU to encode inputs. A better option might be to introduce bi-directional GRU, which helps in content interaction between sentences, by fusing both past and future sentences to be used.

1. Currently we are using GRU to encode inputs. A better option might be to introduce bi-directional GRU, which helps in content interaction between sentences, by fusing both past and future sentences to be used.
2. A fifth module can be introduced, called Summarizing module, which reduces the number of vectors that the episodic memory has to deal with. This can be used to create a summary of the passage, similar to how humans first skim through the document to find relevant parts.

REFERENCES

REFERENCES

1. This project was based on:
   `https://arxiv.org/abs/1506.07285`

2. Our repository for the project:
   `https://github.com/gsaket/DMN-NLP`