The SmartChatBot

Pranoy Kovuri
UIN: 626007676
ECEN 765: Machine Learning with Networks
kpranoy@tamu.edu

***Abstract*— The idea of a QA system is to extract information (sometimes passages, or spans of words) directly from documents, conversations, online searches, etc., that will meet users information needs. Rather than make the user read through an entire document, QA system prefers to give a short and concise answer. Nowadays, a QA system can combine very easily with other NLP systems like chatbots, and some QA systems even go beyond the search of text documents and can extract contextual information and perform other deep semantic tasks; for example, document summarization. In this work, a neural network with a recurrent attention model is used. Memory Network is an end to end model which can be trained using stochastic gradient descent, hence it requires less supervision.**

Keywords— Memory Networks, Stochastic Gradient Descent

I. Introduction And Motivation

My motivation to do this work stems from two domains. The former from the health care industry. U.S health care spending grew 5.8 percent in 2015, reaching $3.2 trillion or $,990 per person. Mental health alone is a 200-Billion-dollar industry. Almost 50% percent of college students self-report symptoms of depression and anxiety, and there are long wait times for counselling service in universities. Here Anti-Depression chat-bots like Woebot can help. The later comes from a domain of expert systems. It would be ideal chat bot which could answer any question on any given topic. This would help people understand complex things within short periods.

Most of the NLP problems can be considered as a question answering problem, the paradigm is simple: we issue a query, and the machine provides a response. By reading through a document, or a set of instructions, an intelligent system should be able to answer a wide variety of questions. Memory networks show promising context understanding and reasoning capabilities in Textual Question Answering (Textual QA). Experiments were conducted on a public Textual Question Answering dataset (Facebook bAbI dataset) with supervision from labels of supporting facts.

In this project we have adopted a end-to-end Memory Network model[2]. The initial model is implemented without LSTMs and the sentence representation is calculated using simple addition of word vectors. The later model is implemented using RNNs and LSTMs and trained end-to-end, and hence requires significantly less supervision during training, making it more generally applicable in realistic settings.

A. *Memory Networks*

Most machine learning models today do not use memory, an element that is one of perks of modern day computer as well as humans. LSTMs are one of the few neural networks which have a memory element. The central idea is to combine the successful learning strategies developed in the machine learning literature for the inference with a memory component that can be read to. (written to in the future work). The model is then trained to learn how to operate effectively with the memory component.

II. Dataset

A. *Babi DataSet from Facebook*

We used public textual QA dataset by Facebook – 10k English dataset. (Here 10K is the sample size of the task) [1]. This dataset includes 20 types of logical reasoning tasks marked from QA1 to QA20, such as QA16: basic induction, QA17: positional reasoning, and QA19: path finding. An example from QA16 is illustrated in Fig. 1. Every sample contains facts, a question, an answer, and labels of supporting facts. There are several other tasks in this dataset including one-supporting fact based, two-supporting fact based, named entity recognition etc.

Facts	*supporting*	+	Question	→	Answer
Lily is a swan.					
Bernhard is a lion.	*Yes* 2				
Greg is a swan.					
Bernhard is white.	*Yes* 3				
Brian is a lion.	*Yes* 1	+	What color is Brian?	→	White
Lily is gray.					
Julius is a rhino.					
Julius is gray.					
Greg is gray.					

Fig. 1 *A sample from Facebook bAbI dataset.*

B. *SQuAD Dataset from Stanford:*

Recently, the Stanford Question Answering dataset (SQuAD) [4], which is orders of magnitude larger than all previous hand-annotated datasets and has a variety of qualities that culminate in a natural QA task. A sample from the dataset can be seen above in Fig. 2. SQuAD has the desirable quality that answers are spans in a reference document. This constrains answers to the space of all possible spans.

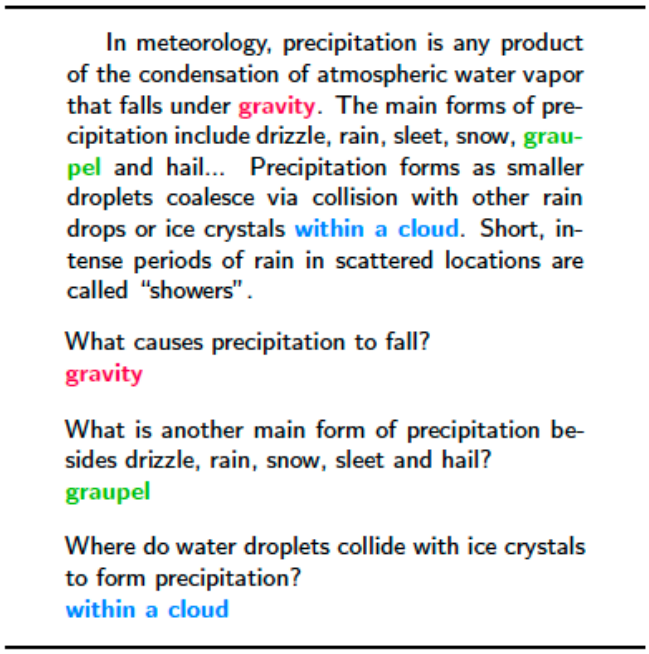


Fig. 2. Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.

III. Preprocessing

A. *Word Processing*

The raw text data is processed and the words are converted to one-hot encoded vectors using vocabulary in the overall corpus. The converted one-hot vectors are then converted into dense vectors of fixed size. Word2vec is a model which transforms the words into vector space based on their embedding. Word2vec models are learned using shallow neural network using either skip-gram or continuous bag of words (CBOW) methods. This unsupervised model learns the contextual similarities between the words. Skip-gram, CBOW, and GloVe (or any other word2vec variant) are pre-trained word embeddings which can be set as the weight of an embedding layer. The weights in the embedding layer are initialized by these pre-trained vectors. The weights are then not updated by the gradient descent, since here are considering pre-trained vectors.

B. *Sentence Processing*

Sentence Embeddings are originally constructed using just additions of the corresponding word embeddings of the words in the sentence. This works because of the simplicity of our dataset, since it is synthetically designed. Although the above doesn't work without exclusively numbering each sentence, this help the neural network identify which sentence to shift its attention to. In the later stage, we used a second representation that encodes the position of words within the sentence.

$$m_i = \Sigma_j \, l_j \, .Ax_{ij} \qquad (1)$$

This takes the form given by the equation (1) where is an element-wise multiplication. Lj is a column vector with the structure. l_{kj} = (1- j/J) - (k/d)(1-2j/J) (assuming 1-based indexing), with J being the number of words in the sentence, and d is the dimension of the embedding. This sentence representation, which we call position encoding (PE), means that the order of the words now affects m_i. The same representation is used for questions, memory inputs and memory outputs.

IV. Tasks

In this project, we mainly focused on two tasks Single Supporting Facts and Two Supporting Facts.

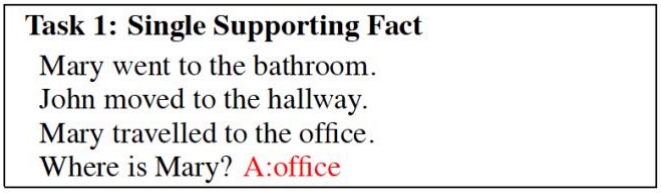


Fig. 3. A Single Support Fact type question answer task

Single Supporting Fact Task 1 consists of questions where a previously given single supporting fact, potentially amongst a set of other irrelevant facts, provides the answer. We first test one of the simplest cases of this, by asking for the location of a person, e.g. "Mary travelled to the office. Where is Mary?". This kind of task was already employed in [5]. It can be considered the simplest case of some real-world QA datasets such as in [6].

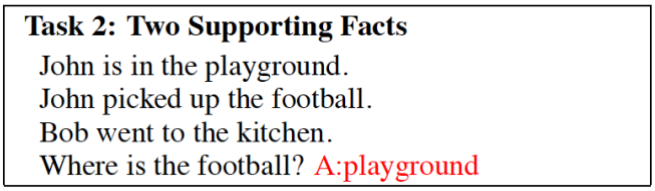


Fig. 4 A Two Supporting Fact Type question answer task

Two or Three Supporting Facts A harder task is to answer questions where two supporting statements have to be chained to answer the question, as in task 2, where to answer the question "Where is the football?" one has to combine information from two sentences "John is in the playground" and "John picked up the football". Again, this kind of task was already used in [5]. Similarly, one can make a task with three supporting facts, given in task 3, whereby the first three statements are all required to answer the question "Where was the apple before the kitchen?".

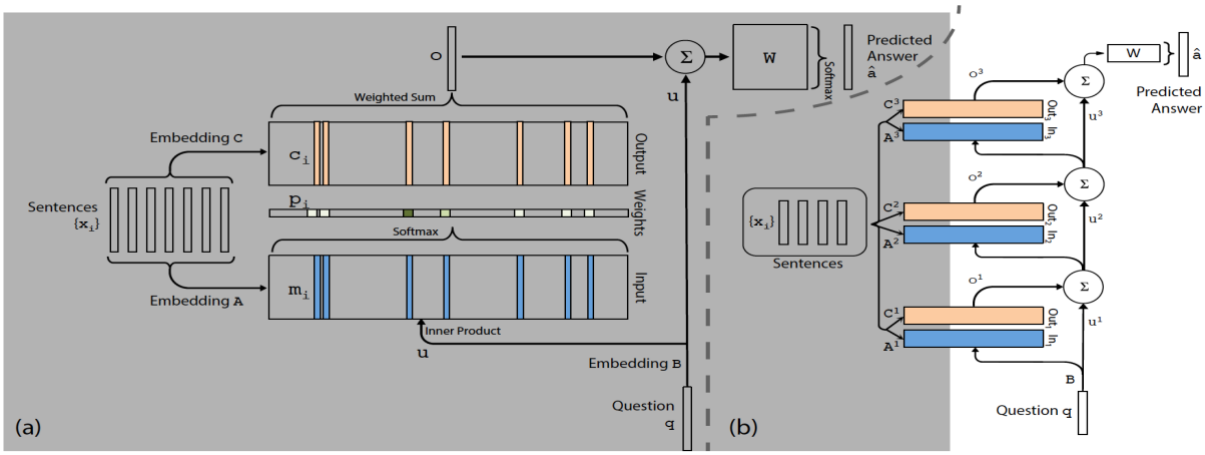


Fig. 5 (a): A single layer version of our model. (b): A three-layer version of our model. In practice, we can constrain several of the embedding matrices to be the same

V. MODEL

A. *MemNNs - Memory Networks*

MemNNs are a class of models which have been shown to perform well in QA. They work by a "controller" neural network performing inference over the stored memories that consist of the previous statements in the story.

In the first version, we use only one hop of inference. This can be seen in the Fig. 5(a). An Attention mechanism based on cosine similarity between the question and all the sentences in the corresponding story. For example, in the following figure after we preprocess the sentences and the query we multiply the both to get the attention for a particular sentence. These are then passed through a softmax layer to get the corresponding weights for each sentence. Then a different embedding representation is calculated for each sentence and the previously calculated weights are multiplied to respective sentences and added together to get a weighted sum of all the sentences. This vector actually knows the information about the answer. Now we pass this vector to a fully connected neural network to get the answer. This whole thing form one hop of the network. The fully connected neural network is trained in a supervised way with the correct answers as labels. To summarize:

- For each story, we take the dot product of every sentence embedding with that story's query embedding. This gives us a list of number proportional to how similar each sentence is with the query.
- We pass this vector of dot products through a softmax function to return a list of scalars that sum to one and tell us how similar the query is to each sentence.
- Next, we construct a second, separate, embedding function for the sentences.
- We then take the weighted average of these embeddings, using the softmax outputs as weights.
- Finally, we pass this weighted average though a dense layer and classify it w/a softmax into one of the words in the vocabulary.

The above mechanism suffices for one-supporting fact kind of tasks. Although for the two-supporting type of tasks we use 2 hops. The output of the fully connected neural network obtained from the above is now used as attention scores for the second layer.

B. *MemN2N - End to End Memory Networks*

In this type of model the sentence embeddings are formed using LSTMs to find the sentence embedding and the complete network is trained in an end to end fashion by minimizing standard cross-entropy between the predicted answer and correct answer. This is only way this network differs from the previous MemNNs is in the sentence embedding and the training phase, everything else is the same including the architecture.

We now extend our model to handle K hop operations. The memory layers are stacked in the following way:

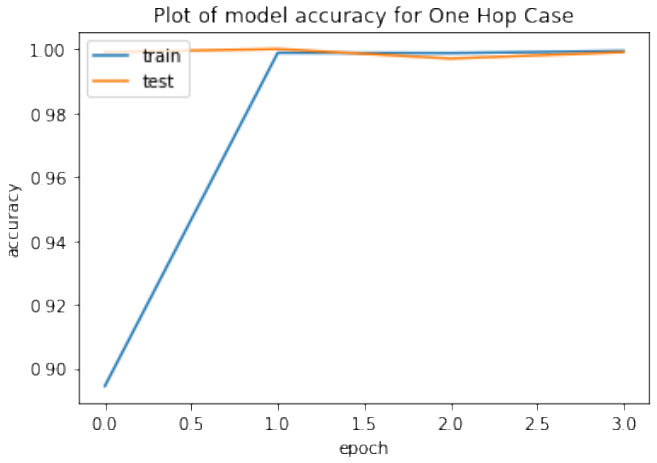
- The input to layers above the first is the sum of the output o^k and the input u^k from the layer k.
- Each layer has its own embeddings matrices A^k, C^k, used to embed the inputs $\{x_i\}$. We restrict these to the same values, i.e. $A^1 = A^2 =:: = A^k$ and $C^1 = C^2 =:: = C^k$

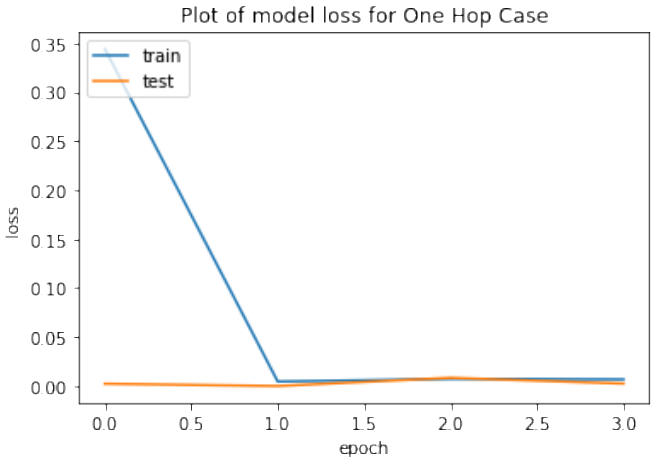
VI. TRAINING DETAILS

A. *First Model MemNNs*

For the one hop case we used an embedding dimension of 20, since the vocab size is only 32. Since this is just training a simple feed forward neural network, there isn't much complication involved. We use an RMSProp optimizer with loss function Sparse Categorical Cross Entropy. We train the system using batching since we have 10000 training stories with a batch size of 32. Four epochs are used for training. The learning rate was 0.01. The results can be seen in the following picture.

```
Train on 10000 samples, validate on 1000 samples
Epoch 1/4
0s - loss: 0.3443 - acc: 0.8945 - val_loss: 0.0023 - val_acc: 0.9990
Epoch 2/4
0s - loss: 0.0046 - acc: 0.9988 - val_loss: 3.1547e-06 - val_acc: 1.0000
Epoch 3/4
0s - loss: 0.0073 - acc: 0.9987 - val_loss: 0.0084 - val_acc: 0.9970
Epoch 4/4
0s - loss: 0.0067 - acc: 0.9994 - val_loss: 0.0025 - val_acc: 0.9990
```



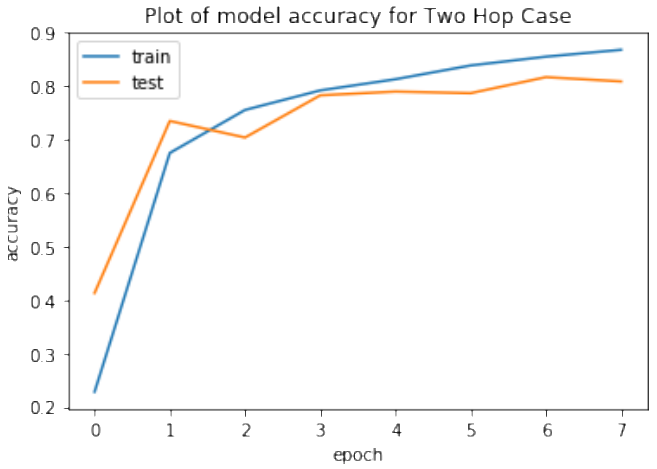



```
([['0:', 'Mary', 'went', 'back', 'to', 'the', 'hallway', '.'],
  ['1:', 'Daniel', 'went', 'back', 'to', 'the', 'bedroom', '.'],
  ['3:', 'Sandra', 'moved', 'to', 'the', 'bathroom', '.'],
  ['4:', 'Sandra', 'journeyed', 'to', 'the', 'hallway', '.'],
  ['6:', 'Mary', 'went', 'back', 'to', 'the', 'bedroom', '.'],
  ['7:', 'Mary', 'went', 'back', 'to', 'the', 'garden', '.']],
 ['Where', 'is', 'Mary', '?'],
 'garden')
```

Fig. 6a Training details single supporting fact case using MemNN, 6b Accuracy as a function of epochs, 6c Loss as a function of epochs, 6d A sample prediction by the network

For the two hop case we used an embedding dimension of 30. We again use an RMS prop optimizer with sparse categorical cross entropy. We faced a lot difficulty fitting this model and finally ended with a batch size of 32 and a learning rate of 0.005 and ran it for 8 epochs. The results can be seen in the following picture.

```
Train on 10000 samples, validate on 1000 samples
Epoch 1/8
6s - loss: 1.8319 - acc: 0.2281 - val_loss: 1.5191 - val_acc: 0.4130
Epoch 2/8
5s - loss: 0.8945 - acc: 0.6751 - val_loss: 0.7426 - val_acc: 0.7350
Epoch 3/8
6s - loss: 0.6686 - acc: 0.7556 - val_loss: 0.7736 - val_acc: 0.7040
Epoch 4/8
7s - loss: 0.5826 - acc: 0.7922 - val_loss: 0.7372 - val_acc: 0.7830
Epoch 5/8
6s - loss: 0.5222 - acc: 0.8131 - val_loss: 0.5666 - val_acc: 0.7900
Epoch 6/8
6s - loss: 0.4772 - acc: 0.8389 - val_loss: 0.6351 - val_acc: 0.7870
Epoch 7/8
5s - loss: 0.4463 - acc: 0.8553 - val_loss: 0.5651 - val_acc: 0.8170
Epoch 8/8
6s - loss: 0.4167 - acc: 0.8682 - val_loss: 0.5908 - val_acc: 0.8090
```




```
  ['84:', 'Daniel', 'took', 'the', 'apple', 'there', '.'],
  ['85:', 'Daniel', 'travelled', 'to', 'the', 'hallway', '.'],
  ['87:', 'Sandra', 'journeyed', 'to', 'the', 'bathroom', '.'],
  ['88:', 'Daniel', 'left', 'the', 'milk', 'there', '.'],
  ['90:', 'Daniel', 'went', 'to', 'the', 'kitchen', '.'],
  ['91:', 'Daniel', 'went', 'back', 'to', 'the', 'bathroom', '.']],
 ['Where', 'is', 'the', 'milk', '?'],
 'hallway')
```

Fig. 7a Training details for two supporting fact case using MemNN, 7b Accuracy as a function of epochs, 7c Loss as a function of epochs, 7d A sample prediction by the network

B. *Second Model - MemN2N*

In this case we used a joint model on all bAbI tasks. The following parameters were used:

- epochs: 60
- hops: 3
- embedding_size: 40

The training results can be seen in the following picture.

Task	Training Accuracy	Validation Accuracy	Testing Accuracy
1	1.0	0.99	0.999
2	1.0	0.84	0.849
3	0.99	0.72	0.715
4	0.96	0.86	0.851
5	1.0	0.92	0.865
6	1.0	0.97	0.964
7	0.96	0.87	0.851
8	0.99	0.89	0.898
9	0.99	0.96	0.96
10	1.0	0.96	0.928
11	1.0	0.98	0.93
12	1.0	0.98	0.982
13	0.99	0.98	0.976
14	1.0	0.81	0.877
15	1.0	1.0	0.983
16	0.64	0.45	0.44
17	0.77	0.64	0.547
18	0.85	0.71	0.586
19	0.24	0.07	0.104
20	1.0	1.0	0.996

Fig. 8 Various accuracies on different tasks for our joint MemN2N model

VII. Results

For the first model, in the one hop case we achieved a pretty high accuracy and low loss owing to the fact of simple fact and one supporting fact. The model predicted almost all of the answers correctly. An instance can be seen in the figure 6d. For the second model, we had a pretty challenging endeavor during the optimization process, but finally achieved a pretty good accuracy metric of 80% on validation set. This is low since this is more difficult compared to the single support fact type. This approach can be extended to n-supporting factor problems by doing n hops.

For the second model, we used a joint model for all the 20 tasks in the bAbi dataset. Hence we designed a metric: For a task to pass it has to meet 95%+ testing accuracy.

Pass: 1,6,9,10,12,13,15,20.

Joint training results are from 10 repeated trails of the joint model across all tasks. The performance of the single model whose validation accuracy passed the most tasks (>= 0.95) is shown in the table on the left.

VIII. Future work

The performance of our model on Squad data set is poor and we intend to improve on this in the future. Inspired by this course project, we intend to work more on this project in the future using Dynamic Memory Networks [8] and Differential Neural Computer [3]. This model can learn to use semantic and episodic mimicking mammals brain memory to answer questions about complex, structured data, including artificially generated stories, family trees.

IX. Conclusion

We have shown that with a memory a machine learning mechanism for reading the memory can be successfully trained on diverse tasks in question answering. Since the model can learn end to end there is no explicit subversion needed. However, there is much to do. Our model did not perform well on Squad dataset and failed on several tasks.

X. References

[1] https://research.fb.com/downloads/babi/

[2] J. Weston, S. Chopra, and A. Bordes. Memory networks. In International Conference on Learning Representations (ICLR) , 2015.

[3] https://www.nature.com/articles/nature20101.pdf

[4] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. In Empirical Methods in Natural Language Processing (EMNLP), 2016.

[5] Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. CoRR , abs/1410.3916, 2014.

[6] Fader, Anthony, Zettlemoyer, Luke, and Etzioni, Oren. Paraphrase-driven learning for open question answering. In ACL , pp. 1608–1618, 2013.

[7] Sukhbaatar, Sainbayar, szlam, arthur, Weston, Jason, and Fergus, Rob. End-to-end memory networks.

[8] Xiong, Caiming, Merity, Stephen, and Socher, Richard. Dynamic memory networks for visual and textual question answering. In ICML , 2016.