TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

Evaluating the Performance of Keras Implementation of MemNN Model for Simple and Complex Question Answering Tasks

R. Poonguzhali[1], Dr. K. Lakshmi[2]
Computer Science and Engineering
[1]Research Scholar, Periyar Maniammai Institute of Science and Technology.
[2] Professor, Periyar Maniammai Institute of Science and Technology.

Article Info
Volume 82
Page Number: 9620 - 9629
Publication Issue:
January-February 2020

Article History
Article Received*: 18 May 2019*
Revised*: 14 July 2019*
Accepted*: 22 December 2019*
Publication*: 12 February 2020*

Abstract:
Question Answering (QA) system is a field of Natural language processing, which allows users to ask questions using the natural language sentence and return a brief answer to the users rather than a list of documents. Memory networks are capable of reasoning with inference components combined with a long-term memory component and they learn how to use these two components in an efficient way to predict answers from the story text for a specific question. This work intends to evaluate the performance of an earlier keras implementation of memory network (MemNN) model and compare its performance with three standard deep learning models RNN, LSTM and GRU.

In this work, we implement a Keras implementation of MemNN model based question answering systems and evaluate their performance with a simple and complex question answering tasks from bAbI dataset. We will study the performance of training and testing with suitable metrics and find the difference in performance in the two question answering tasks.

Keywords: *NLP, QA, Deep learning, MemNN, Memory Networks, RNN, LSTM, GRU bAbI Tasks .*

I. INTRODUCTION

Natural Language Processing makes the computer to understand the language of humans. The human language is highly ambiguous due to the nature of its structure. The ambiguities are either Lexical, Syntax level, Referential etc.,[1]. Question Answering (QA) is Natural Language Processing task which generates automatically short and precise answers to natural language questions given by users. The user need not worry about the system as they enter the question in their own language. The QA system answer the question based on the keywords traditionally. But nowadays, as the technology grows the system answer the question based on the facts or stories on that subject . So most of the problems in artificial intelligence and Natural language processing can be represented as question answering problems [8].

The QA system may be generally divided as Closed Domain Question Answering (CDQA) and Open Domain Question Answering (ODQA). CDQA systems extract the answers based on stored knowledge base for the query given by users. ODQA answer questions from any domain from unstructured data [3].

The four main approaches of QA systems are Rule-Based, Statistical Approach, Machine Learning and Deep Learning.

Due to availability of large question answer datasets, data driven methods are proven to be very powerful in Question Answering [11]. The idea is the data is used to drive the system other than methods in QA systems. The data can be taken from rich linguistic resources like dictionaries, Word Net etc., [12].

Rule-based mechanism was one of the most significant methods of QA systems. These systems used heuristic rules that look for lexical and semantic clues in the question [13]. These rules are used for interpretation of question classification. A major drawback of rule-based question answering systems are it should be written manually and also this consumes lot of time [14].To write the rules manually, an in-depth knowledge about the structure of language is needed.[15].

With the advancement in technologies and available of text repositories, statistical approaches came into existence for QA systems. These approaches essentially need sufficient amount of data for statistical learning but once it learned appropriately. It gives better results than other competing approaches.

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

Furthermore, the learned statistical method can be easily adapted to any language. Thus this approach is adapted to various stages of QA. A major drawback of statistical approaches is those consider each term separately and failed to understand the semantics of phrases. [16] [17].

Machine learning approaches can learn to understand linguistic features without explicitly annotated. Machine learning approaches achieves competitive accuracy compared to approaches which depend on handwritten, deterministic rules and algorithms The advantages of machine learning algorithms are highly scalable and their ability to optimize over time[18].

Memory Networks produce state of the art results in Natural Language Processing task [19]. One such architecture is dynamic memory networks which gives high accuracy in variety of language tasks.

Question answering (QA) has a long history within natural language processing, going back to the 1960s and 1970s, with systems such as Baseball and Lunar. Baseball require human to communicate with computers in natural language. It read simple questions about baseball games. But it avoid complex questions with multiple dependent clauses or logical connectives (e.g., and, or, not). A later system called Lunar aimed at querying the chemical analysis data on lunar rock and soil composition as a result of Apollo moon missions. As Text Retrieval conference (TREC) introduced QA track on Open domain question answering, QA systems become popular from 1999. Even though Current QA systems deal with simple factual questions, more system needed to answer complex questions. One such question is temporal question [7].

Natural language processing (NLP)

Natural language processing (NLP) is a component of Linguistics and Artificial Intelligence which makes computer to understand words or statements of human languages. It helps the users to communicate with their own natural languages. The two components of NLP are Natural Language Understanding and Natural Language Generation. The most significant tasks of NLP are question answering, machine translation, information extraction, automatic summarization, sentimental analysis, Named entity recognition, Optical character recognition etc. [1]

Language independent NLP

Language independent system must work equally well across all languages. If an algorithm is created for one language it could be extended to another language. Thus language independence is considered to be one of the machine-learning approaches to NLP [10].

Deep Learning

Deep learning is a field of machine learning which allows computational models that has multiple processing layers to learn vectorized data. Deep learning makes major advancement in solving problems. Deep learning has given tremendously good results for different tasks in natural language understanding like question answering and language translation, topic classification, sentiment analysis [22].

The bAbI tasks for NLP Research

In [4], the authors propose a list of tasks under common framework which are generated for testing text understanding and reasoning abilities. It is a set of 20 tasks and each task tests a unique aspect of text and reasoning and hence tests different capabilities of learning models. This data can be used for testing text understanding and reasoning models.

Any QA model can be trained and tested on each task and they give good results in testing data. Based on Weston et al. (2014), training set consists of true answers and also relevant sentences to answer the questions or supporting facts [4]. Refer [4] for more information about these tasks.

This dataset consists of several different tasks:

- en/ denotes the tasks in English. It contains 1000 examples.
- hn/ denotes the tasks in Hindi. It contains 1000 examples.
- shuffled/ denotes the same tasks with shuffled letters. This is the only task which is not readable by humans. It contains 1000 examples.
- en-10k/ shuffled-10k/ and hn-10k/ denotes the same tasks in all the three formats. It consists of 10,000 training examples.

The file format for each task is as follows:

```
ID text
ID text
ID text
ID question[tab]answer[tab]supporting fact IDS.
...
…
```

About this work

This work intends to evaluate the performance of an earlier keras implementation of memory network (MemNN) model and compare its performance with three standard deep learning models like RNN, LSTM and GRU. In this work, we implement a Keras implementation of MemNN model based question answering systems and evaluate their performance with a simple and complex question answering tasks from bAbI dataset.

We will study the performance of training with low and high number of training samples and with simple and complex QA tasks. We will study the performance of training and testing with suitable metrics and find the difference in performance in the two question answering tasks.

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

II. MODELING

Modeling a QA system

Training a Typical QA System

The following diagram illustrates the typical process of training a questing answering system. Using a word-vector dictionary, the training story texts, question texts and their corresponding answers texts will be vectorized and a deep learning network will be trained with the vectorized training data.

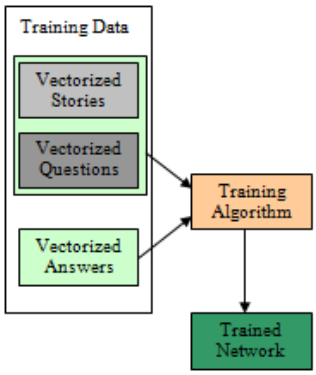


Figure 1. The Training Process

Testing a Typical QA System

The following diagram illustrates the typical process of testing or validating a questing answering system. The test story texts, and question texts will be vectorized and fed in to the trained network and the network will predict the possible answer vectors. The actual answer test from the vectorized answers will be created using reverse lookup in the word-vector dictionary.

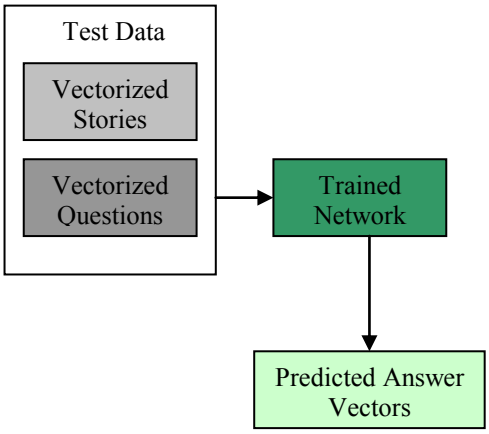


Figure 2. The Testing Process

The MemNN[6]

In [6] the authors present a new class of models called memory networks that combine large memory with learning component that can read and write to it. Memory networks has inference components which is combined with a long-term memory component. The long-term memory can be read a story and answer questions from it.

A memory network consists of a memory m which is an array of objects either vectors or strings indexed by m_i and four (potentially learned) components I, G, O and R as follows:

- I: denotes input feature map which map the input into feature representation.
- G: denotes generalization which updates old memories, given the new input.
- O: denotes output feature map which gives a new output.
- R: denotes response which converts the output into the preferable response format desired.

The steps of this model are as follows:

- Convert x to an internal feature representation I (x).
- Update memories mi given the new input: $m_i = G(m_i, I(x), m), \forall i$.
- Compute output features o given the new input and the memory: o = O(I (x), m).
- Finally, decode output features o to give the final response: r = R(o).

This process is applied at both train and test time, if there is a distinction between such phases, that is, memories are also stored at test time, but the model parameters of I, G, O and R are not updated. Memory networks cover a wide class of possible implementations. The components I , G, O and R can potentially use any existing ideas from the machine learning literature, e.g., make use of your favorite models (SVMs, decision trees, etc.).

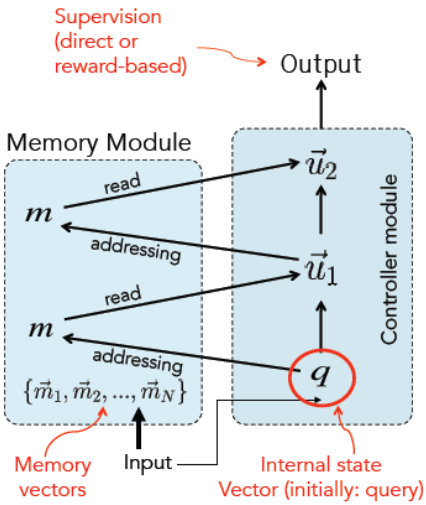


Figure 3. MemNN Model

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

MemNN Implemention

I (input): converts to bag-of-word-embeddings x.

G (generalization): stores x in next available slot m_N.

O (output): Loops over all memories k=1 or 2 times:

1st loop max: finds best match m_i with x.

2nd loop max: finds best match m_J with (x, m_i).

The output o is represented with (x, m_i, m_J).

R (response): ranks all words in the dictionary given o and returns best single word.

Training phase :

Training is performed with a margin ranking loss and stochastic gradient descent (SGD). Specifically, for a given question x with true response r and supporting sentences m_{o_1} and m_{o_2} (when $k = 2$), we minimize over model parameters U_O and U_R:

Minimize:

$$\sum_{\bar{f}' \neq m_{O_1}} \max(0, \gamma - s_O(x, m_{O_1}) + s_O(x, \bar{f}')) +$$

$$\sum_{\bar{f}' \neq m_{O_2}} \max(0, \gamma - s_O([x, m_{O_1}], m_{O_2}]) + s_O([x, m_{O_1}], \bar{f}')) +$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_{O_1}, m_{O_2}], r) + s_R([x, m_{O_1}, m_{O_2}], \bar{r})) +$$

……..(1)

Where:

S_O is the matching function for the Output component.

S_R is the matching function for the Response component.

x is the input question.

m_{O1} is the first true supporting memory (fact).

m_{O2} is the first second supporting memory (fact).

r is the response

True facts and responses m_{O1}, m_{O2} and r should have higher scores than all other facts and responses by a given margin.

$\bar{f}$, $\bar{f}'$ and $\bar{r}$ are all other choices than the correct labels, and γ is the margin. At every step of SGD we sample $\bar{f}$, $\bar{f}'$, $\bar{r}$ rather than compute the whole sum for each training example

In the case of employing an RNN for the R component of MemNN (instead of using a single word response as above) the last term was replaced with the standard log likelihood used in a language modeling task, where the RNN is fed the sequence [x, o_1 , o_2 , r]. At test time the model output its prediction r given [x, o_1 , o_2]. In contrast the absolute simplest model, that of using $k = 1$ and outputting the located memory m_{o_1} as response r, would only use the first term to train.

The MemNN Modeled with Keras

The following diagram illustrates the layers of the MemNN network designed in keras for question answering system.

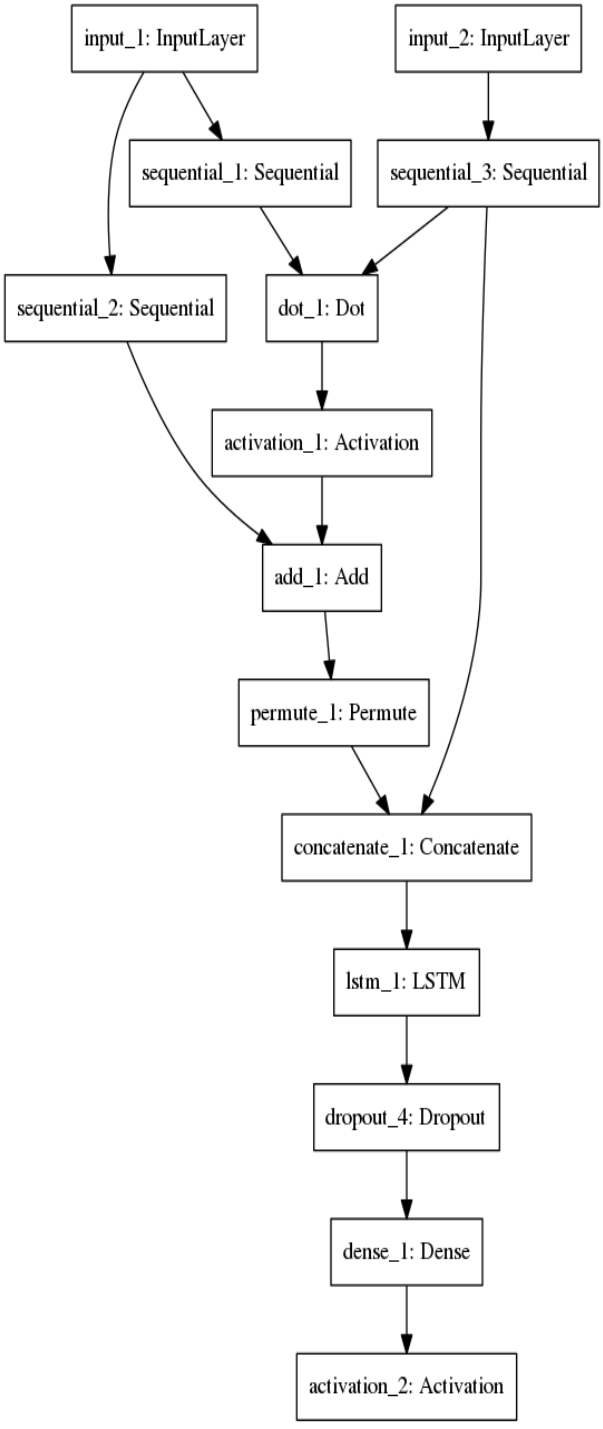


Figure 4. The MemNN Layers

TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

III. THE RESULTS AND DISCUSSION

The layer information of different deep learning networks(Ex: bAbI tasks ID- 20)

Simple RNN

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 69)	0	
input_2 (InputLayer)	(None, 8)	0	
embedding_1 (Embedding)	(None, 69, 64)	2688	input_1[0][0]
embedding_2 (Embedding)	(None, 8, 64)	2688	input_2[0][0]
simple_rnn_1 (SimpleRNN)	(None, 32)	3104	embedding_1[0][0]
simple_rnn_2 (SimpleRNN)	(None, 32)	3104	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 64)	0	simple_rnn_1[0][0] simple_rnn_2[0][0]
dense_1 (Dense)	(None, 42)	2730	concatenate_1[0][0]

Total params: 14,314, Trainable params: 14,314, Non-trainable params: 0

LSTM

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 69)	0	
input_2 (InputLayer)	(None, 8)	0	
embedding_1 (Embedding)	(None, 69, 64)	2688	input_1[0][0]
embedding_2 (Embedding)	(None, 8, 64)	2688	input_2[0][0]
lstm_1 (LSTM)	(None, 32)	12416	embedding_1[0][0]
lstm_2 (LSTM)	(None, 32)	12416	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 64)	0	lstm_1[0][0] lstm_2[0][0]
dense_1 (Dense)	(None, 42)	2730	concatenate_1[0][0]

Total params: 32,938, Trainable params: 32,938, Non-trainable params: 0

GRU

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 69)	0	
input_2 (InputLayer)	(None, 8)	0	
embedding_1 (Embedding)	(None, 69, 64)	2688	input_1[0][0]
embedding_2 (Embedding)	(None, 8, 64)	2688	input_2[0][0]
gru_1 (GRU)	(None, 32)	9312	embedding_1[0][0]
gru_2 (GRU)	(None, 32)	9312	embedding_2[0][0]
concatenate_1 (Concatenate)	(None, 64)	0	gru_1[0][0] gru_2[0][0]
dense_1 (Dense)	(None, 42)	2730	concatenate_1[0][0]

Total params: 26,730, Trainable params: 26,730, Non-trainable params: 0

MemNN

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 69)	0	
input_2 (InputLayer)	(None, 8)	0	
sequential_1 (Sequential)	multiple	2688	input_1[0][0]
sequential_3 (Sequential)	(None, 8, 64)	2688	input_2[0][0]
dot_1 (Dot)	(None, 69, 8)	0	sequential_1[1][0] sequential_3[1][0]
activation_1 (Activation)	(None, 69, 8)	0	dot_1[0][0]
sequential_2 (Sequential)	multiple	336	input_1[0][0]
add_1 (Add)	(None, 69, 8)	0	activation_1[0][0] sequential_2[1][0]
permute_1 (Permute)	(None, 8, 69)	0	add_1[0][0]
concatenate_1 (Concatenate)	(None, 8, 133)	0	permute_1[0][0] sequential_3[1][0]
lstm_1 (LSTM)	(None, 32)	21248	concatenate_1[0][0]
dropout_4 (Dropout)	(None, 32)	0	lstm_1[0][0]
dense_1 (Dense)	(None, 42)	1386	dropout_4[0][0]
activation_2 (Activation)	(None, 42)	0	dense_1[0][0]

Total params: 28,346, Trainable params: 28,346, Non-trainable params: 0

TEST Engineering & Management

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

IV. PERFORMANCE EVALUATION

We carried out all our experiments and evaluations on a laptop with Intel Core i7 Processor with 16GB RAM. We didn't use any GPU/TPU during training phase. We developed all the code in Python language (ver 3.5) with Keras and Tensorflow.

Performance Metrics Used:

Accuracy

Accuracy is a metric which evaluates classification models. The performance of classification model is identified using confusion matrix calculated for the corresponding classifier. The entries in matrix compute and all other metrics. All of these performance measures are easily obtainable for binary classification problems. The measure needs to be chosen based on the classifier. Hard classifiers produce an outcome $g(x) \in \{1,2,\ldots,k\}$.On the other hand, Soft classifiers produce quantities on which a cutoff can be applied to find g(x). Traditionally, accuracy is defined as the average number of correct predictions:

$$Accuracy = \frac{1}{n}\sum_{k=1}^{|G|}\sum_{x:g(x)=k} I(g(x)=\hat{g}(x)) \quad \ldots\ldots\ldots\ldots\ldots\ldots(2)$$

where I is the indicator function, which returns 1 if the classes match and 0 otherwise.

Loss

Loss or Mean Squared Error (MSE) or Mean Squared Deviation (MSD) measures the sum of errors in each iteration. The error will be the average squared difference in the target value and the real value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

Results with Low Number of Training Samples

Training and Testing Parameters:

Training Data Directory : data/tasks_1-20_v1-2/en

Total Training Samples :1000

Total Testing/Validation Samples :1000

No Epochs :100

Training Batch Size : 32

Evaluating Performance with Small Training Data

The networks were trained with 1000 samples and tested with another 1000 samples. The following table shows the testing performance with 1000 samples of Task ID-1. (complex task).

Table 1. The Results with The bAbI tasks ID: 1
(with 1000 training samples and 1000 testing samples)

Model	No Trainable Parameters	Time Taken for Training (s)	Accuracy	Loss
SimpleRNN	10,454	175	0.48	1.77
LSTM	29,078	377	0.45	1.72
GRU	22,870	416	0.49	1.38
MemNN	24,494	74	0.44	1.54

The following table shows the testing performance with 1000 samples of Task ID-20 (easy task).

Table 2. The Results with The bAbI tasks ID: 20
(with 1000 training samples and 1000 testing samples)

Model	No Trainable Parameters	Time Taken for Training (s)	Accuracy	Loss
SimpleRNN	14,314	188	0.94	0.61
LSTM	32,938	402	0.97	0.21
GRU	26,730	453	0.94	0.31
MemNN	28,346	105	0.93	0.28

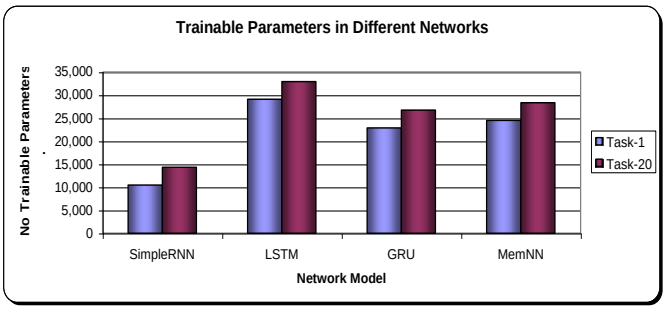


Figure 5. The Comparison of No. Parameters
(Task 1 vs 10k Task 20)

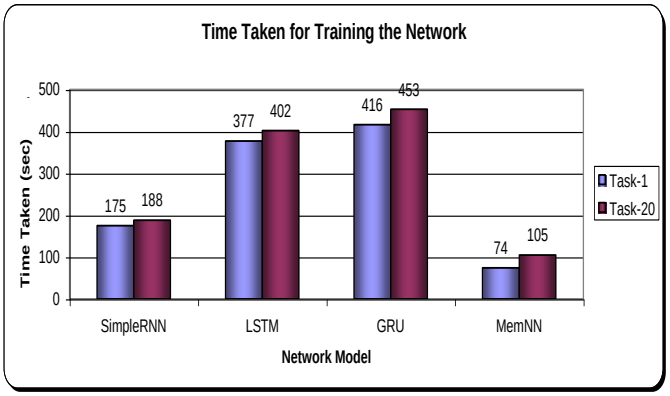


Figure 6. The Comparison of Training Time
(Task 1 vs 10k Task 20)

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

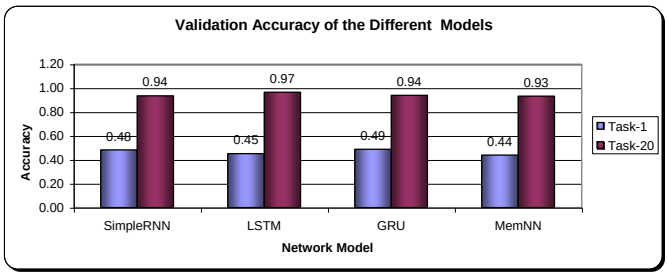


Figure 7. The Comparison of Accuracy
(Task 1 vs 10k Task 20)

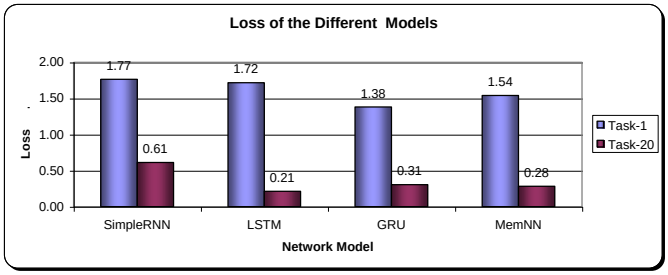


Figure 8. The Comparison of Loss/MSE
(Task 1 vs 10k Task 20)

As shown in the above results with 1000 training samples and 1000 testing samples of bAbI Task-1, all the four models gave almost equal performance. Even though the some of the previous evaluations with MemNN shows better result (accuracy =1.0) on bAbI Task-1, in our evaluations, we didn't get higher accuracy near 1.0. This may be because of lower number of training samples used to train MemNN.

Results with High Number of Training Samples

So, we try to train all the networks with 10,000 samples of bAbI Task-1 and verify the improvements in performance in all the four models. The following shows the training parameters used and the corresponding results.

Training and Testing Parameters:

Training Data Directory : data/tasks_1-20_v1-2/en

Total Training Samples :10,000

Total Testing/Validation Samples :1000

No Epochs :100

Training Batch Size : 32

The following table shows the testing performance with 10,000 samples of Task ID-1 (complex task).

Table 3. The Results with The bAbI tasks ID: 1
(with 1000 training samples and 10,000 testing samples)

Model	No Trainable Parameters	Time Taken for Training (s)	Accuracy	Loss
SimpleRNN	10,454	1421	0.51	1.25
LSTM	29,078	3017	0.47	1.35
GRU	22,870	3490	0.49	1.26
MemNN	24,750	576	0.95	0.16

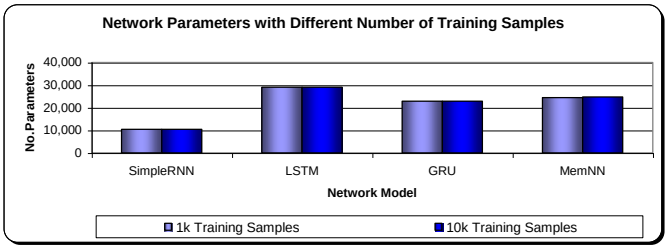


Figure 9. The Comparison of No. Parameters
(1k vs 10k Training Data)

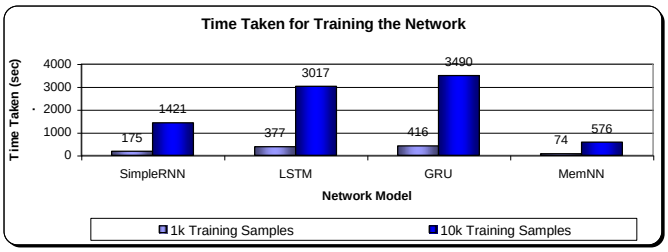


Figure 10. The Comparison of Training Time
(1k vs10k Training Data)

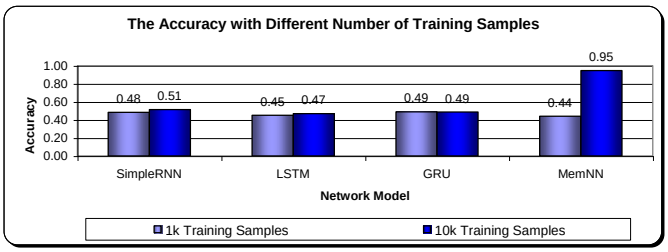


Figure 11. The Comparison of Accuracy
(1k vs 10k Training Data)

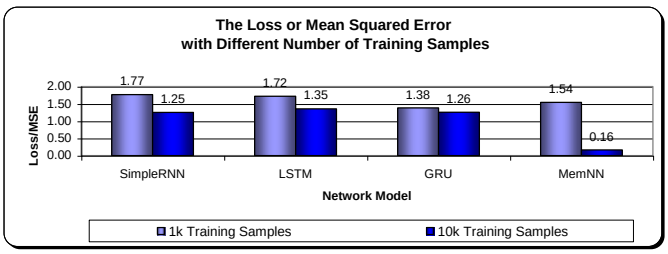


Figure 12. The Comparison of Loss/MSE
(1k vs10k Training Data)

The above results clearly shows that the increase in number of training samples significantly improved the performance of MemNN but does not improve the performance of the other models significantly. This shows that the MemNN based models are capable of handling some of the complex question answering tasks.

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

V. CONCLUSION

The results of our evaluation clearly shows that all the four evaluated models performed almost equal in terms of accuracy if the question answering task is complex (if the models are trained with low number of training samples such as 1000 samples). If the question answering task is complex, and more training samples (ex: 10000 samples) are available, then MemNN model will provide very good accuracy than other three standard models. Most importantly, in all kinds of tasks, the performance of MemNN was very good in terms of training time as well as accuracy.

From the results we can clearly understand that the performance of MemNN is getting improved significantly with the use of high number of training samples. But the performance of all the other three classic models were not at all getting improved significantly with the use of high number of training samples. So, in our future works, we will study more about these memory networks and their performance on different kinds of QA tasks.

VI. ACKNOWLEDGEMENT

We thank the Management of Periyar Manimmai Institute of Science and Technology for providing us the facilities to complete this work successfully.

VII. REFERENCES

[1] Deeksha Dwivedi, Mahendra Singh Sagar, "Overview of Natural Language Processing", 4th International Conference on System Modeling & Advancement in Research Trends (SMART), 2015.

[2] Tom Young, Devamanyu Hazarika, Soujanya Poria, Erik Cambria,"Recent Trends in Deep Learning Based Natural Language Processing", arXiv:1708.02709v8 [cs.CL], 25 Nov 2018.

[3] Kolomiyets Oleksander and Marie-Francine Moens, A survey on question answering technology from an information retrieval perspective, in: Journal of Information Sciences volume 181, 2011.

[4] Jason Weston, Antoine Bordes, Sumit Chopra, Tomas Mikolov, Alexander M. Rush, "Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks", arXiv:1502.05698v10 [cs.AI] 31 Dec 2015, (conference paper at ICLR 2016).

[5] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, "End-To-End Memory Networks", arXiv:1503.08895v5 [cs.NE] 24 Nov 2015.

[6] Jason Weston, Sumit Chopra & Antoine Bordes, "Memory Networks", arXiv:1410.3916v11 [cs.AI], 29 Nov 2015, (conference paper at ICLR 2015).

[7] Poonguzhali, K Lakshmi, "Temporal Question Answering System: A Survey", JETIR June 2019, Volume 6, Issue 6, ISSN-2349-5162.

[8] Yashvardhan Sharmaa, Sahil Guptaa, "Deep Learning Approaches for Question Answering System", International Conference on Computational Intelligence and Data Science (ICCIDS 2018).

[9] K.S.D. Ishwari, A.K.R.R.Aneeze, S.Sudheesan, H.J.D.A. Karunaratne, A. Nugaliyadde, Y.Mallawarrachchi, "Advances in Natural Language Question Answering: A Review", arXiv:1904.05276 [cs.CL], Apr 2019.

[10] Emily M. Bender, "Linguistically Na¨ive != Language Independent: Why NLP Needs Linguistic Typology", Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics, pages 26-32, Athens, Greece, 30 March, 2009.

[11] L. Kodra and E. Kajo, "Question Answering Systems: A Review on Present Developments, Challenges and Trends", International Journal of Advanced Computer Science and Applications, vol. 8, no. 9, 2017

[12] E. Brill, J. Lin, M. Banko, S. Dumais and A. Ng, "Data-Intensive Question Answering", Trec.nist.gov, 2018.

[13] H. Madabushi and M. Lee, "High Accuracy Rule-based Question Classification using Question Syntax and Semantics", Aclweb.org, 2018.

[14] E. Riloff and M. Thelen, "A Rule-based Question Answering System for Reading Comprehension Tests", 2018.

[15] S. Humphrey and A. Brownea, "Comparing a Rule Based vs. Statistical System for Automatic Categorization of MEDLINE Documents According to Biomedical Specialty", 2018.

[16] S. K. Dwivedia and V. Singh, "Research and reviews in question answering system," in Proceedings of International Conference on Computational Intelligence: Modeling Techniques and Applications, 2013, pp. 417 - 424 .

[17] D. Cohn, Z. Ghahramani and M. Jordan, "Active Learning with Statistical Models", Journal of Artificial Intelligence Research, vol.4, 1996.

[18] X. Li and D. Roth, "Learning Question Classifiers", Dl.acm.org, 2018.

[19] Raghuvanshi, A., & Chase, P., "Dynamic Memory Networks for Question Answering".

[20] Ramesh Sharda et.el, "Business Intelligence and Analytics: Systems for Decision Support", Copyright© 2015, 2011, 2007 by Pearson Education, Inc., ISBN 10: 0-13-305090-4, ISBN 13: 978-0-13-305090-5

[21] Elvis, "Deep Learning for NLP: An Overview of Recent Trends", https://medium.com/dair-ai

[22] LeCun,Y.,Bengio,Y.&Hinton,G."Deep learning" N*ature* **521,**436–444,2015, https://doi.org/10.1038/nature14539.

[23] R Poonguzhali, K Lakshmi, "Evaluating the Performance of Recurrent Neural Network based Question Answering System with Easy and Complex bAbI QA Tasks" Manuscript submitted for publication.

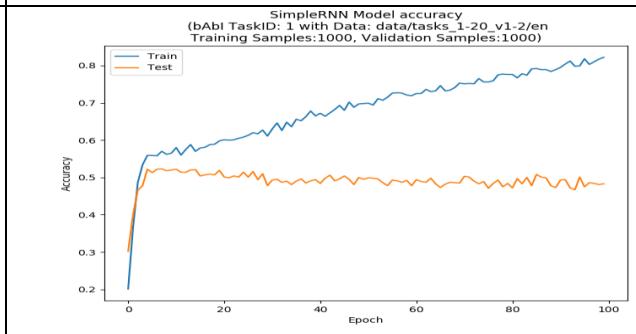
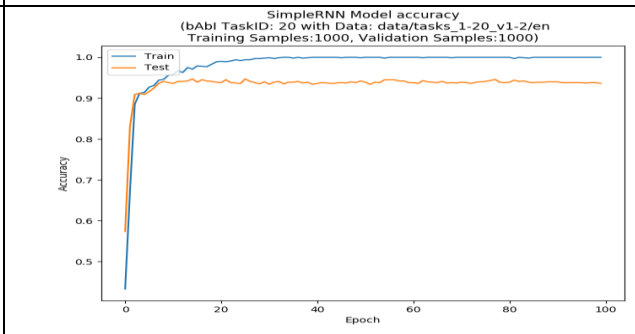
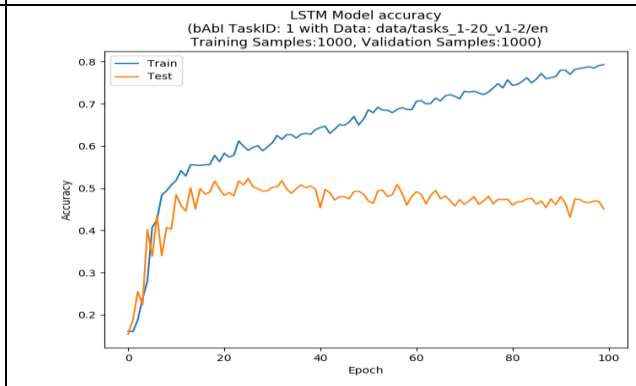
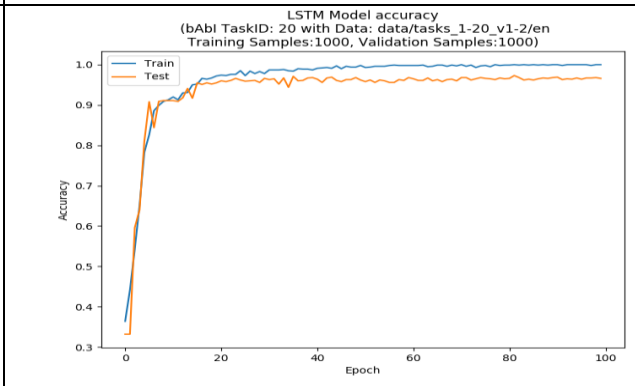
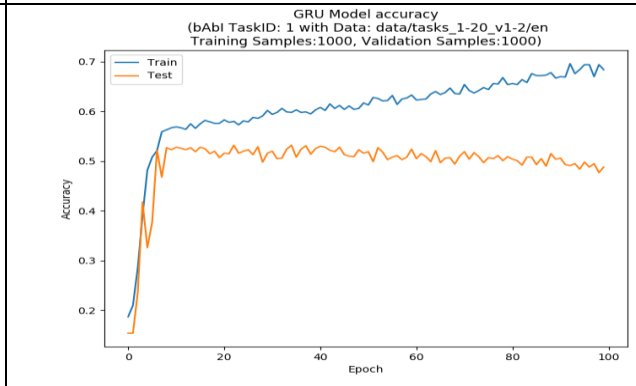
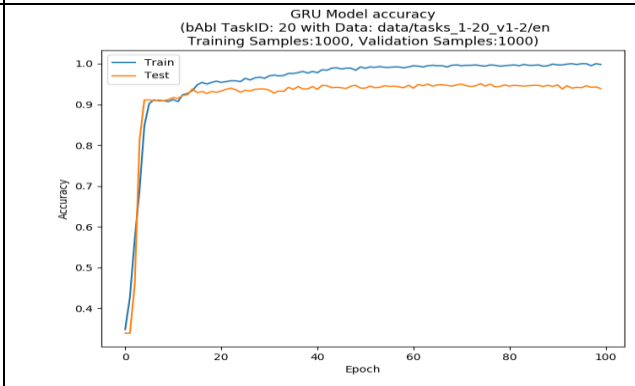
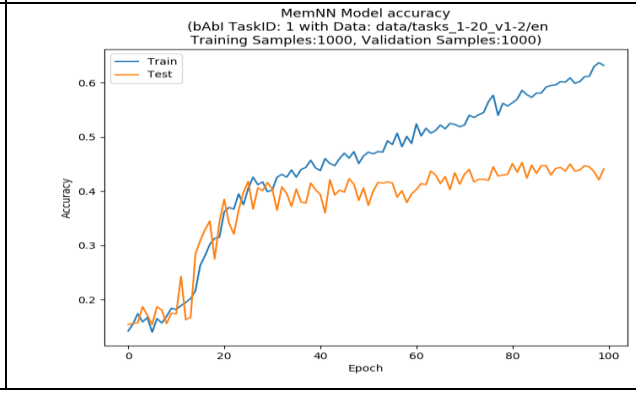
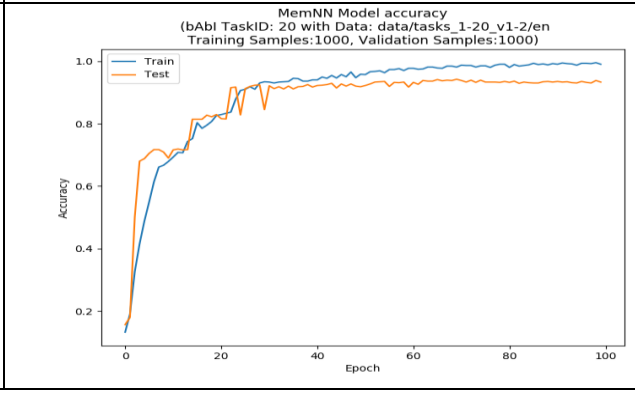
[24] R Poonguzhali, K Lakshmi, "Analysis on the Performance of Some Standard Deep Learning Network Models for Question Answering Task" Manusript accepted for publication.

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

VIII. ANNEXURE

Performance with 1000 Training Samples and 1000 Testing Samples of QA Task-1 and 20

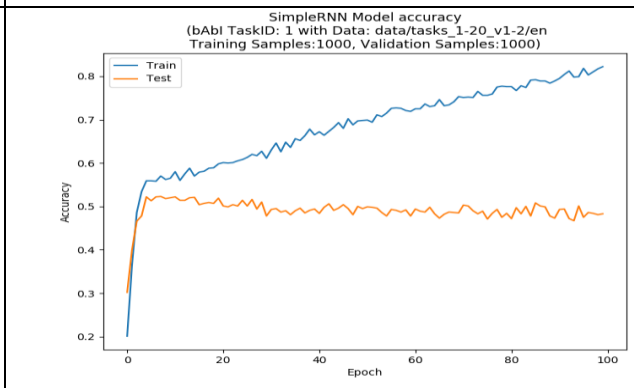
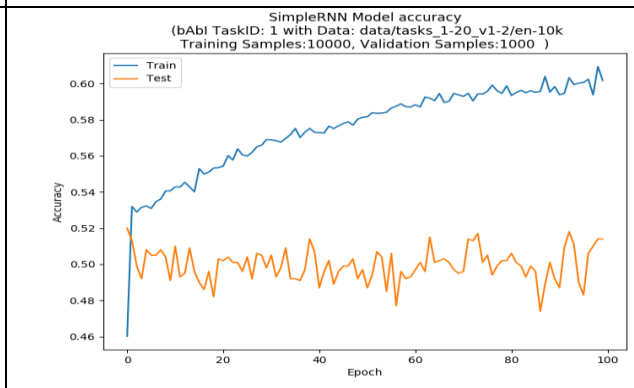
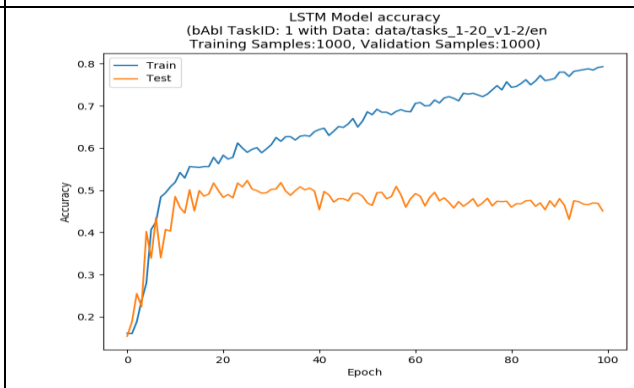
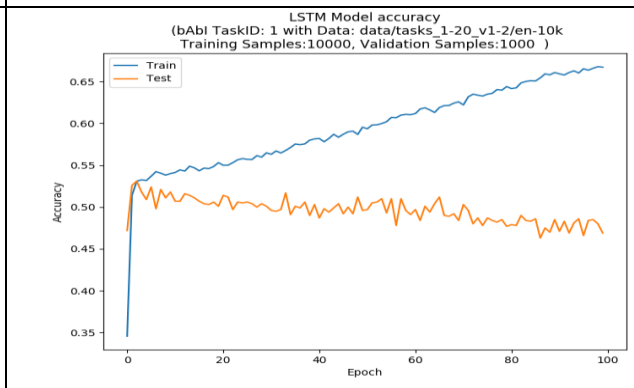
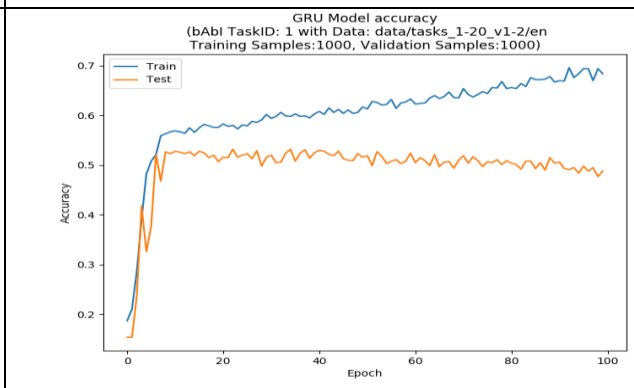
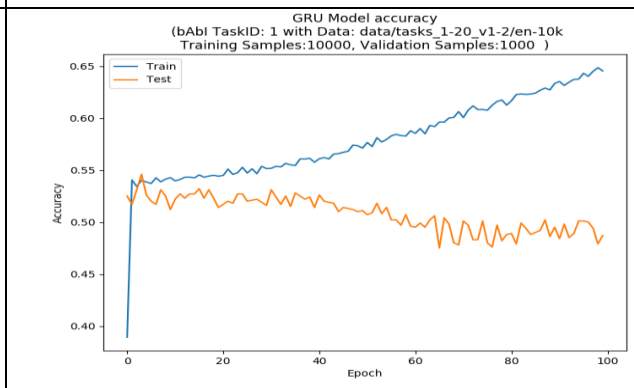
The following three set off graphs clearly shows the complex nature of task 1 QA data. With Task-1 data, with all the three models, the training is not getting improving after few epochs. But in the case of task-20, all the three models are providing better accuracy above 0.9 after few epochs of training.

	Accuracy with 1000 Training Samples (Task ID-1)	Accuracy with 1000 Training Samples(Task ID-20)
SimpleRNN (1k Samples)		
LSTM (1k Samples)		
GRU (1k Samples)		
MemNN(1k Samples)		

January - February 2020
ISSN: 0193 - 4120 Page No. 9620 - 9629

Performance with 10000 Training Samples and 1000 Testing Samples of Complex QA Task-1

The first column is the same as above- we repeat it for the purpose of easy comparison with the other output (of Accuracy with 10,000 Training Samples). From the following graphs we can clearly understand that the performance of MemNN is getting improved significantly with the use of high number of training samples. But the performance of all the other three classic models were not at all getting improved significantly with the use of high number of training samples.

	Accuracy with 1000 Training Samples (Task ID-1)	Accuracy with 10,000 Training Samples (Task ID-1)
SimpleRNN		
LSTM		
GRU		
MemNN