Codalab Username: stevekuo4, hsinya, yanghy

Machine Comprehension Using Dynamic Recurrent Neural Network and Gated Recurrent Unit

Yi-Hong Kuo
Adobe
stevekuo4@gmail.com

Hsin-Ya Lou
Chemistry Department
Stanford University
hsinya@stanford.edu

Hsiang-Yu Yang
Chemistry Department
Stanford University
yanghy@stanford.edu

Abstract

Machine comprehension (MC) is one of the most challenging problems in natural language processing. Using deep learning techniques is considered of being a promising way to build a reliable machine learning model for solving MC. The previous exciting datasets, however, are difficult to train and evaluate deep learning models due to their limitation in both size and complexity, respectively. Recently, the Stanford Question Answering Dataset (SQuAD), a new released dataset including a huge number of contexts and related questions generated by crowdsourcing, provides a reliable dataset to help develop and evaluate more MC models. By using SQuAD, we developed an end-to-end neural-network architecture to approach the MC problems. The architecture includes dynamic recurrent neural network that are able to process questions and contexts with different lengths, and also includes gated recurrent unit (GRU) that can "memorize" the previous part of word sequences during processing.

1 Introduction

Machine comprehension (MC) of text is considered as one of the most difficult task in natural language processing. The task is to make machine to read a given paragraph and answer several questions related to the given paragraph. This task not only requires the machine to understand both paragraphs and questions, but also need to find the logical relation between them. In recent years, several context-question-answer dataset have been created to evaluate the performance of different machine comprehension models. For example, RCTest [Richardson et al., 2013] includes 500 stories with 4 multiple choice questions for each story. However, the size of this dataset is not enough to build deep neural network models. Many MC methods based on this dataset are using fabricated feature or embedding background knowledge [Sachan et al., 2015] [Wang et al., 2016].

In order to overcome the limitation of the previous datasets, the Stanford Question Answering dataset (SQuAD) has been developed by humans through crowdsourcing to make the dataset more realistic [Rajpurkar et al., 2016]. The dataset includes 500+ articles with over 100k questions, which has more questions than previous datasets with about two orders of magnitude. In addition, instead of multiple choice questions, answers are designed to be subsequences within the articles, make it even more challenge to predict the answers.

In this paper, we are proposing an end-to-end deep neural network architecture for machine comprehension model of text based on the SQuAD dataset. With the fact that the answer of a given context-question pair in SQuAD is a subsequence in the context, we need to predict both the start and end indexes in the context. Since different samples may have different word lengths, we choose dynamic recurrent neural network to encode both question and passage into representation matrix. Many papers use long-short term memory (LSTM) to keep the information while processing the questions and contexts. However, compared with LSTM, gated recurrent unit

(GRU) is shown to have similar performance while has less parameter and higher training efficiency [Kumar et al. 2016]. Here we combine dynamic memory network with GRU (DMN-GRU) to build the architecture.

In the following parts, we will give more details on the MC task in SQuAD dataset, and describe our DMN-GRU model. Finally we will discuss the evaluation results of our model on the SQuAD dataset.

2 SQuAD Task Description

SQuAD task includes answering few questions with a given passage that containing the answers (an example shown in Table 1). We can consider every training sample in SQuAD is a set of tuples (P, Q, A), in which P represents the passage with m words: $P = (W_1^p, W_2^p, ..., W_m^p)$, Q represents the question with n words: $Q = (W_1^q, W_2^q, ..., W_n^q)$, and A represents the answer including two numbers represent the starting (a_s) and end index (a_e) in the passage: $A = (a_s, a_e)$, $1 \le a_s \le a_e \le N$. Thus, we can consider the task as estimate the conditional probability of two indexes with given question and passage $Pr(A|Q,P)$.

Table 1: Examples of SQuAD passage-question-answer pairs showing that the answer to each question is either a sequence of words or a single token in the given passage.

Passage	Question	Answer
With Rivera having been a linebacker with the Chicago Bears in Super Bowl XX, and Kubiak replacing Elway at the end of the Broncos' defeats in Super Bowls XXI and XXIV, this will be the first Super Bowl in which both head coaches played in the game themselves.	In what Super Bowl did Rivera play?	Super Bowl XX
	What team did Rivera play for in Super Bowl XX?	Chicago Bears
	What position did Rivera play in Super Bowl XX?	linebacker

3 Model Description

Our deep neural network model for machine comprehension consists of several layer. We have test two different neural network structures (illustrated in **Figure 1**): a baseline model based on a simplified version of Dynamic Coattention Networks [Xiong et al., 2017], and an enhanced model with dynamic memory.

The high level structure of both models is:

1. Word Embedding Layer: Maps each word to a vector using pre-trained word embedding model
2. Passage Encoding Layer: Encode the passage word embedding matrix using recurrent neural network with GRU as cell structure.
3. Question Encoding Layer: Encode the question word embedding matrix using recurrent neural network with GRU as cell structure.
4. Dynamic Memory: A process which combines an attention mechanism and a recurrent network to update an internal memory. This process will iterate several times. In each iteration, the importance of each context word is re-evaluated based on the question representation and the last internal memory. This mechanism provides a way to use reasoning and inference to answer the question. This module is not included in the baseline model.
5. Modeling Layer: With the encoded passage and question matrices (and the internal memory) as input, the modeling layer uses attention mechanism to see which parts of the inputs need to focus on, and generates an attention contexts. A vector representing both the question matrix and passage matrix is then generated.
6. Decoding Layer: Transform the above mentioned representation to log probabilities of being the boundary pointers for each context word.

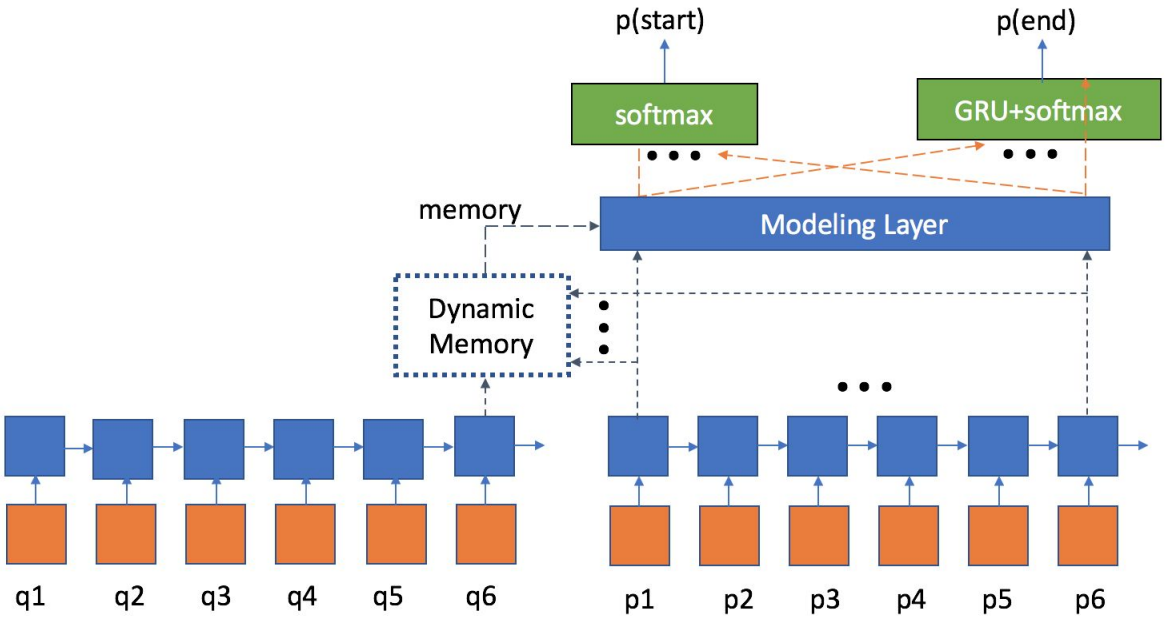


Figure 1. An overview of the model.

3.1 Word Embedding Layer

The word embedding layer maps each word in questions and passages into a vector space. Here we use GloVe [Pennington et al., 2014], a pre-trained word vector database to fix the word embedding for each word. The output of this layer are two matrices: Question Matrix (Q) and Passage Matrix (P): $Q \in R^{d \times m}$, $P \in R^{d \times n}$. The d represents the state size of the word embedding.

3.2 Passage Encoding Layer

We use bidirectional recurrent neural network with GRU [Chung et al., 2014] to process passage matrix P in order to get the temporal relation between the words in the passage. Two output matrices in both directions are concatenated along the hidden-state axis to output the passage representation (R_P) $R_P \in R^{m \times 2h}$. The h represents the state size of the hidden layer.

3.3 Question Encoding Layer

We also use bidirectional recurrent neural network with GRU to process the question matrix Q. However, instead of concatenating the output matrices, we concatenate the final states from both directions of the RNN to output the question representation (R_Q) $R_Q \in R^{1 \times 2h}$.

3.4 Dynamic Memory Layer

Once get the representation matrices R_P and R_Q, the dynamic memory layer will take the matrices as input and iterates them [Kumar et al., 2016]. This layer includes an attention mechanism and a recurrent network in which it will keep updating its memory (structure illustrated in **Appendix Figure 1**). For every iteration, the new episodic memory m_i is generated by GRU with previous memory m_{i-1} and an episode e_i, $m_i = GRU(e_i, m_{i-1})$, with the question representation as the initialized membrane state $m_0 = R_Q$. We use a gating function as our attention mechanism. For every iteration, the mechanism takes three inputs: final state of the passage matrix

after passing through a recurrent neural network C_t, the question representation R_Q, and previous memory m_{i-1}.

First, the scoring function G will takes the inputs to generate a scalar score at state. Several feature vectors are defined as z matrix to represent the similarities between S_P, R_Q and m_{i-1}:

$$z(C_t, m_{i-1}, R_Q) =$$
$$[C_t, m_{i-1}, R_Q, C_t \circ R_Q, C_t \circ m_{i-1}, |C_t - R_Q|, |C_t - m_{i-1}, |, C_t^T W R_Q, C_t^T W m_{i-1}] \quad (1)$$

Second, the z matrix will be feed into the function G which contains two layer of neural network:

$$G(z) = \sigma(W^{(2)} tanh(W^{(1)} z + b^{(1)}) + b^{(2)}). \quad (2)$$

The output of the G function is a scalar, which represents the score at state i g_t^i.
Finally, we use a modified GRU to process the sequence of the input C_1 to C_M weighted by the score g_t^i. The output are the question representation R_Q and the final hidden state of the modified GRU (h_M^i), $h_M^i \in R^{1\times 2h}$:

$$h_t^i = g_t^i \, GRU(C_t, h_{t-1}^i) + (1 - g_t^i) \, h_{t-1}^i \quad (3), \; e^i = h_M^i \quad (4)$$

3.5 Modeling Layer

For the sequence-to-sequence mixing baseline model,
$C_P = Q * \text{softmax}(Q^T P)$ (5),
$C^Q = Q * \text{softmax}(Q^T W P + b)$ (6),
$P = \text{concat}(C_P, P)W + b$ (7)

For the DMN model,
$P_1 = \text{concat}(P \circ R_Q, P)W + b$ (8),
$P_2 = \text{concat}(P \circ m, P)W + b$ (9),
$P = \text{concat}(P_1, P_2)$ (10)

3.6 Decoding Layer

Then we use another linear layer to get the log probabilities of being the starting pointer:

$$Pred_{Start} = P \, W_4 \quad (11)$$

For the ending pointer, another GRU before the linear layer is used to model the log probabilities:

$$Pred_{End} = \text{GRU}(P) \, W_5 \quad (12)$$

After getting the prediction vectors, the *qa_model.search_match* function will find the (start, end) pair with the maximum multiplication. The final predicted answer is the two indices which maximize the conditional probability:
$A_{pred} = arg\,max_{1 \le a_s \le a_e \le N,\, a_e - a_s \le M} [Pr(a_s|Q,P) * Pr(a_e|Q,P)]$. (13)

$Pr(a_s|Q,P)$ and $Pr(a_e|Q,P)$ represent the conditional probability of with given question and passage. Here we add an upper limit (M) on how long the answer can be. M can be tuned to maximize the final metrics on dev dataset after the model is trained.

4 Experiments

4.1 Dataset and experiment setting

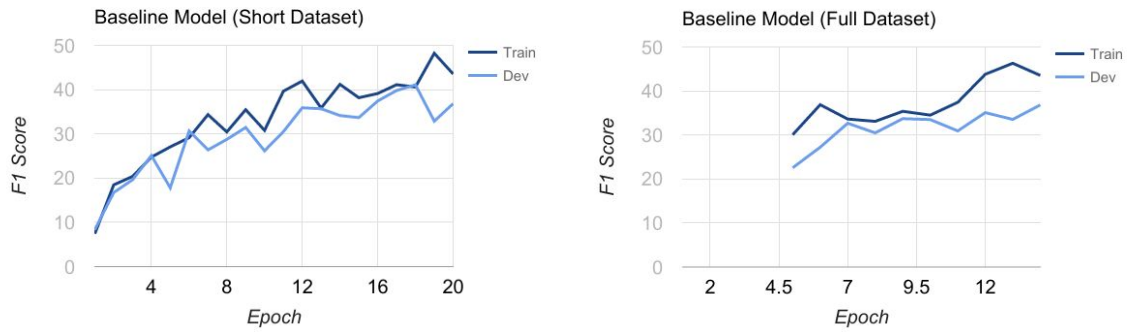
Stanford Question Answering Dataset (SQuAD) v1.1 is used for our experiments. We split the dataset into few parts: a training set (with 81381 question-answer pair), a development set (with 4284 question-answer pair), and a test set. In addition, we also make a mini-training set (⅛ of the training set) to debug our model. The datasets were first be tokenized and mapped to vectors by pre-trained GloVe word embedding to initialize the model input. Words not in the GloVe are set to be zero vectors. The word embedding is fixed during the training.

The model is trained by minimizing the sum of cross-entropy of the $Pred_{Start}$ and $Pred_{End}$. To evaluate the performance of our model, two metrics will be measured: F1 score from comparing the tokens in predicted answers with the tokens in ground truth answers, and the percentage of exact match with the ground truth answers.

4.2 Effect of questions and passages length on model performance

Because most of the passages and questions have relative short length comparing to the maximum length within the dataset (the distribution of passage/question/answer length is shown in **Appendix Figure 2**), it may be beneficial to discard the dataset with long passage and questions. The advantage is that it leads to smaller input dimension that allows us to increase the hidden state size, which is importance for learning, and also faster training time. On the other hand, the final model may performance badly for long passage, so there is a tradeoff between prediction power on the long and show passage.

To test this idea, we create a Short Dataset, where the passages and questions with length above a threshold will be removed. Here we set the threshold to be 400 tokens for passages, 30 tokens for questions based on the histogram . We train 2 baseline models with the same parameters, but using either the Short Dataset or the Full Dataset, where all the data is retained. **Figure 2** shows that the models trained with Short Dataset or the Full Dataset produce similar F1 and EM score. Therefore for the following experiments we only user Short Dataset for training. For evaluating model, we will simply truncate the questions/passages to the threshold instead of discarding the data.



.

Figure 2. The learning performance of the baseline model with short and full dataset. Both models show around 40 of F1 score in both Train and Dev dataset after training.

4.3 Results of Baseline Model

To compare with our model, we implement a baseline model without dynamic memory module (structure shown in **Appendix Figure 3**). For training the baseline model, we tried both stochastic gradient descent (SGD) and Adam optimizer. In the initial test run, both optimization methods lead to similar results, but Adam optimizer, once it is tuned, converge faster than SGD. Therefore we chose Adam optimizer for all the following experiments.

In **Figure 3** shows F1 on training and dev dataset at different epoch. The F1 on training dataset keep increasing while the F1 on development data reaches maximum at epoch 10 and decreases afterward. Therefore it is necessary to implement early stopping. In this project we use a very simple rule that if the F1 on the development dataset does not increase for 3 epoch, we will stop the training process manually, and select model that produce the maximum F1 on dev set. Also, the gap between train and development dataset, and the fact that dev F1 decrease, implies overfitting. To avoid overfitting we also add dropout on the outputs of every layers.

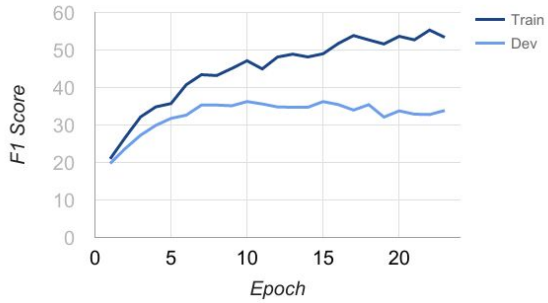


Figure 3. The performance (F1 score) of the baseline model on training dataset and development dataset. The F1 on development dataset reaches maximum at epoch 10.

We tested out various combinations of hidden state size, dropout rate, and maximum answer range (**Table 2.**)

Table 2. Models evaluation with different hyperparameters.

model	Train F1 [%]	Train EM [%]	Dev F1 [%]	Dev EM [%]
Baseline, State size=100, keep_prob = 1.0	51.86	36.85	35.28	22.20
Baseline, State size=100, keep_prob = 1.0, Max answer=10	51.86	36.85	37.86	25.06
Baseline, State size=100, keep_prob = 0.8, Max answer=5	50.27	35.15	44.44	33.47
Baseline, State size=100, keep_prob = 0.8, Max answer=10	50.27	35.15	44.73	32.25
Baseline, State size=100, keep_prob = 0.8, Max answer=20	50.27	35.15	43.89	30.46
Baseline, State size=100, keep_prob = 0.8, Max answer=30	50.27	35.15	42.67	29.08
Baseline, State size=100, keep_prob = 0.8, Max answer=40	50.27	35.15	41.61	27.97
Baseline, State size=100, keep_prob = 0.5, Max answer=10	28.94	17.35	30.79	19.80
Baseline, State size=200, keep_prob = 0.8, Max answer=10	**63.39**	**47.50**	**46.26**	**32.68**
Baseline, State size=300, keep_prob = 0.8, Max answer=10	61.76	44.80	46.05	32.45

4.4 DMN-GRU Model Results

We would like to extend or change the sequence-to-sequence mixing model to see if we can get better results compared with the baseline model discussed before, with the use the dynamic memory network.

Initially, we just used the last memory and question representation as the input for decoder. These two vectors were concatenated and sent to the decoding layer. This initial model performed poorly, as shown in the first two raws in Table 3. Compared with the Baseline model, we found that the dimension of the encoder outputs between this and the Baseline model is different. The Baseline Encoder outputs a passage representation matrix and a co-attention matrix. Both matrices have the dimension (hidden, passage length), which means that they provide information with respect to each position within the passage. However, this initial dynamic memory model only output the final state of both the memory layer and the question representation, which combines all of the hidden information including the position information. Since the information is already messed up, it is impossible to use decoder to get the position information back. Thus, we need to modify our initial model to make sure that we keep the position information. The modified DMN-GRU_v5 model (Code in *encoder_decoder.py*: *DMNEncoderV5* and *DecDMNV1*) has quite differences compared with the initial DMN-GRU model:

In the improved DMN model *DMNEncoderV5*, We did element-wise multiplication between R_Q (or m) and each component vector of P, and mixed P and the products to form a large, combined feature tensor:

$P_1 = \text{concat}(P \circ R_Q, P)W + b,$
$P_2 = \text{concat}(P \circ m, P)W + b,$
$P = \text{concat}(P_1, P_2)$

In **Figure 4**, we can see that the F1 score of training set for the DMN-GRU_v5 increased to 65.4 and development set increased to 42 after 9 epochs, and EM percentage is 29.2% for development set, which is way more better than the initial model and also similar with the baseline model. We also tested different hyperparameters to see how they influence the performance of the DMN-GRU model (**Table 3**).

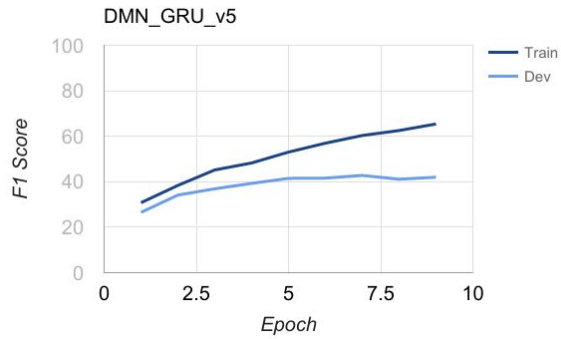


Figure 4. The learning performance of the modified DMN-GRU_v5 model. (ADAM optimizer, Learning Rate = 0.001)

Moreover, we also tried different ways to utilize information generated from the dynamic memory network. For example, in the *DMNSeq2SeqV1* encoder in *encoder_decoder.py*, in addition to mixing of P and Q, we also mix $H = \{h_t^i\}$ and P in the same way and concatenate the two mixed tensor as input of decoder. The result of this model is also shown in Table 3. However, we are still only able to achieve similar performance as the baseline model.

Table 3. Models evaluation with different hyperparameters. In the initial testing, the encoder only outputs two vectors, the question representation R_Q and the last memory m. These two vectors were concatenated and sent to the decoding layer.

model	Train F1 [%]	Train EM [%]	Dev F1 [%]	Dev EM [%]
Initial testing, # memory update = 1, State size=100, keep_prob = 1.0	11.68	4.50	9.24	3.00
Initial testing, # memory update = 3, State size=100, keep_prob = 1.0	12.36	7.50	8.70	3.00
DMN-GRU_v5, # memory update = 1, state size = 200, keep_prob = 0.5	60.68	44.25	38.82	26.01
DMN-GRU_v5, # memory update = 1, state size = 200, keep_prob = 0.8	65.39	49.15	42.00	29.02
DMN-Seq2Seq-GRU, # memory update = 1, state size = 100, keep_prob = 0.8	35.50	21.00	34.85	21.00
DMN-Seq2Seq-GRU, # memory update = 1, state size = 200, keep_prob = 0.8	47.46	32.75	43.18	28.9

5 Conclusion

In this paper, we developed DMN-GRU model for the machine comprehension problem using Stanford Question Answering Dataset (SQuAD). It is found that the DMN-GRU model, achieve better EM percentage at about 29.2%. However, the overall performance is still similar with the baseline model and need to be improved. In the future, we plan to further optimize the hyperparameter to see how well this model could be, and may also improve the structure of the memory module. Ultimately, we hope to use this model to apply to other machine comprehension datasets.

6 Author Contribution

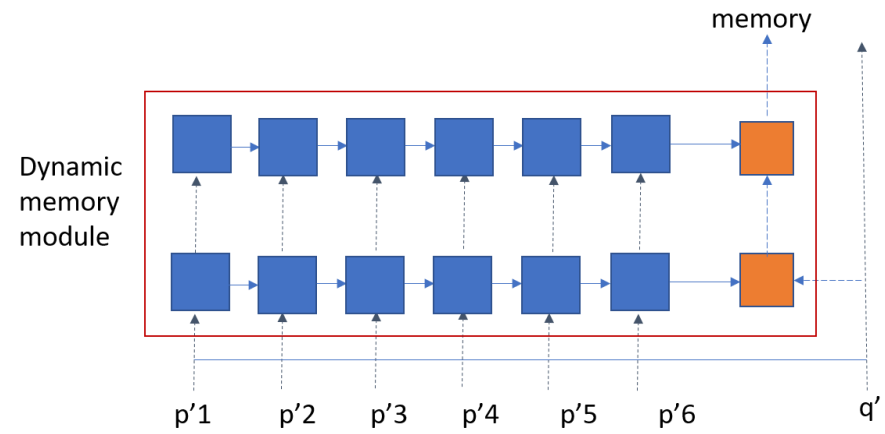
Y.-H. Kuo, H.-Y. Lou and H.-Y. Yang contribute for the model design and initial model implementation. Y.-H. Kuo and H.-Y. Yang contribute for the baseline model and the modified model implementation. H.-Y. Lou contributes for the data collection and paper writing.

Reference

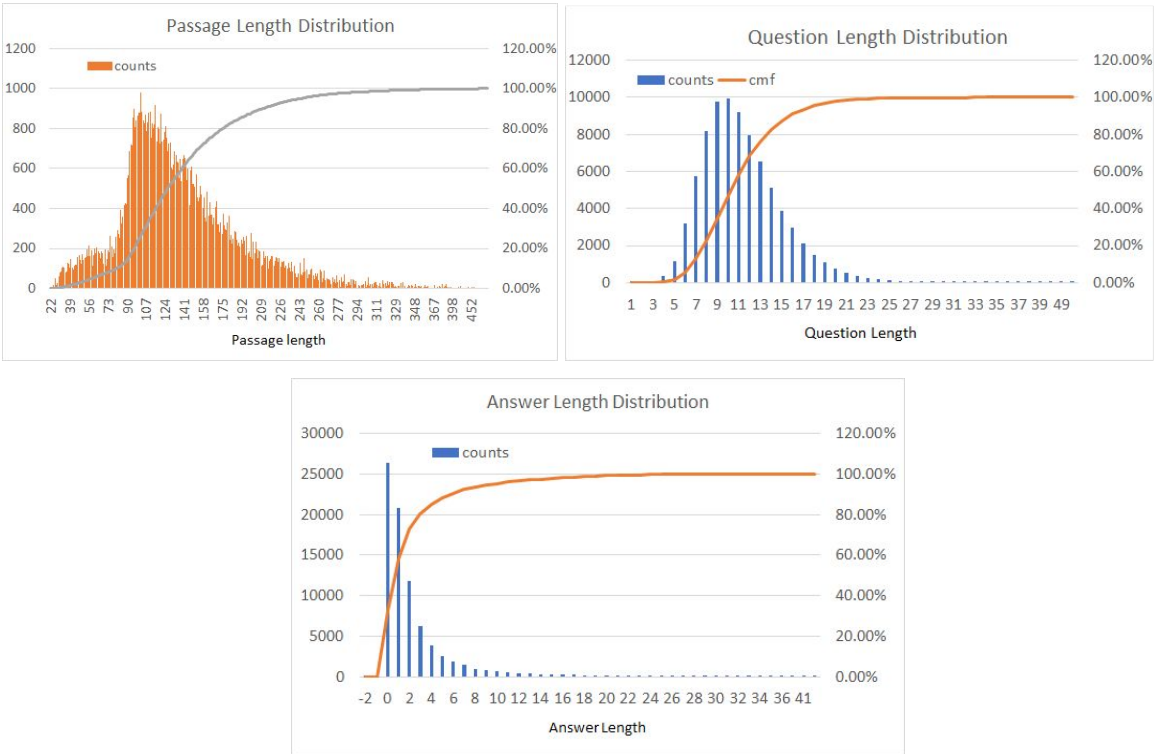
1. Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2013.
2. Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answerentailing structures for machine comprehension. In Proceedings of ACL, pages 239–249.
3. BingningWang, Shangmin Guo, Kang Liu, Shizhu He, and Jun Zhao. 2016a. Employing external rich knowledge for machine comprehension. In Proceedings of IJCAI.
4. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016.
5. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In EMNLP, 2014.
6. Chung, Junyoung; Gulcehre, Caglar; Cho, KyungHyun; Bengio, Yoshua (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling".

7. Ankit Kumar, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, Richard Socher. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing.
8. Caiming Xiong, Victor Zhong, Richard Socher. Dynamic Coattention Networks for Question Answering. In ICLR 2017.
9. Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hananneh Hajishirzi. Bi-Directional Attention Flow for Machine Comprehension. In ICLR 2017.

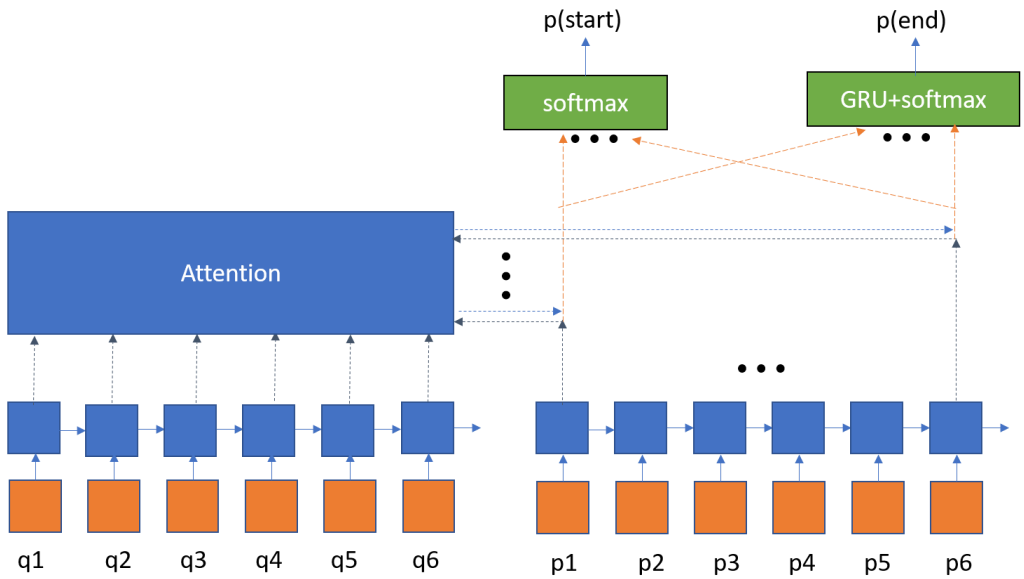
Appendix



Appendix Figure 1. The structure of the dynamic memory module.



Appendix Figure 2. The distribution of the length of passage/question /answer in the SQuAD dataset. For the short dataset ,which set the threshold to be 400/30 for passage/question, about 5% of the samples was removed from the original dataset.



Appendix Figure 3. The structure of the Baseline model.