Short Passages Reading Comprehension and Question Answering

Yuan Yang
Northeastern University
yang.yuan1@husky.neu.edu

Zhe Hu
Northeastern University
hu.zhe@husky.neu.edu

Abstract

Allowing computers to directly answer questions is a big goal of NLP. There are several released datasets for machine reading comprehension task, including Facebook bAbI Tasks, Stanford Question Answering Dataset (SQuAD), and Microsoft MARCO dataset. In this project, we apply two neural network based model, End to End Memory Network and R-NET on the datasets, we compare them and try to improve their performance.

1 Introduction

Machine comprehension and QA (question answering) are important research topics in NLP domain. In this project, we will try to create a model to answer simple questions by reading a short passage. To be specific, our task is defined as follows:

Answer a simple question by reading a short passage, where the question should be clear and not ambiguous, and the answer should be directly given in the passage. During implementation, we limited the upper bound of passage size to be 300 words and limit the upper bound of question size to be 30 words. However, those are not the limitation of the models.

Here is a passage example: "… Andrew went to school at 9 AM and ate a sandwich for lunch and came back home at 5 PM. …" By reading this passage our model should able to answer questions like "When did Andrew come home?" or "What did Andrew have for lunch?" But might not able to answer questions like "Where was Andrew at 1 PM?" or "What did Andrew do today?" or "Did Andrew have lunch at school?"

We think this task design is more closes to the requirements from search engine users. It's better to give a short passage with a highlighted answer, the users can trust the answer more by giving evidence.

In this project, we applied two machine comprehension models, End to End Memory Network and R-NET, on the datasets. We compare both models on the datasets we selected, and implement the model with some modifications to make them perform better and faster.

2 Background and Related Work

QA has always been a research hotspot in NLP, and it has also been used in many applications such as dialog systems. Traditionally, people use IR (information retrieval) and knowledge-based methods to solve QA problems, such as Siri and IBM Watson.

In machine comprehension tasks, we want to answer questions base on the given passage, and the answers are usually part of the passage, which are single words or sub-span of passage. Also, to answer the questions, we do not need extra knowledge but only use the information of the given story. From the setup, the machine is usually given some text fist, like a passage or story. And then based on the given text, the machine is expected to answer the questions.

Recently, with the development of deep learning methods, researchers tend to use neural network based model to solve this problem. Most of the models are based on recurrent neural networks, like LSTM and GRU, which are good to handle the sequence data like text. All these models share some common ideas. Firstly, both passage and questions are usually encoded by encoders, and then some attention mechanisms are used to get passage representations when taking account of the questions. The outputs are usually parts of the passage or the story. Like in End to End Memory Network, the output is a single word chosen from the passage. In Match-LSTM and R-NET the outputs are indexes of the sub-span of the passage. Before this project, the neural network models archived a very good performance in QA competitions like SQuAD Leaderboard. We also decided to try neural network based models in this project.

3 Data

3.1 Data Sets

We mainly use three datasets: bAbI, SQuAD, MARCO. And we also tried to get more data by generating our own datasets.

Facebook bAbI dataset: This dataset is available in English and Hindi, with 1000 questions for training and 1000 for testing. It was designed for QA and text understanding tasks. The dataset is composed of a set of contexts, with multiple question-answer pairs. The vocabulary is very limited, and the form of sentences is quite simple.

Stanford Question Answering Dataset (SQuAD): 100,000 English question-answer pairs on 500 articles. SQuAD consists of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. This dataset fit the definition of our task perfectly.

Microsoft MARCO Dataset: 100,000 English queries along with corresponding passages and answers. This is a great data set because it contains real questions asked by humans on Microsoft search engines. Microsoft also provided more than one passages which might be able to answer the question. If our model can work well on this dataset which means we are close to providing answers to the real search engines.

Extra Data: We parse Wikipedia pages to get more data sets. First, we break pages into paragraphs. If the paragraph has a moderate length (i.e. 200-300 words), then parse the POS tags and try to find some patterns, and generate corresponding questions and answers. For example, for pattern "‧ {NNP} {VBD} ... on {CD}.", if the {CD} also match a time format, we can generate question "When did {NNP} {VB} …?". Wikipedia has sufficient pages, thus the patterns could to be very restricted. But still, human reviewing is needed to select good quality question answer pairs. Considering that these extra data might be quest biased before we have sufficient patterns, we didn't use this dataset as the main dataset.

3.2 Data Preprocessing

Different datasets have differences in detail, but we basically follow same steps to preprocess the data.

Tokenization and indexing: We first tokenize the passages and words to indexes. We use the index in GloVe (glove.840B.300d, words indexes start from 1, the 0 is reserved for UNK).

Drop long passages and questions: We have limitations on the length of the passages, for data exceed the limitation, we just drop them instead of cropping them. We have two reasons: First, trimmed passages might have different meanings. Second, the answer might more likely to be around the center of the trimmed passage, which might end with a biased model. For long questions, we think it's safe to drop it because usually there exists a shorter representation.

Drop answers not directly given in passage: We try to locate the answer in the passage. For SQuAD dataset, the location of the answer is given, but there are some errors, so we still need validation during preprocessing. For bAbI task 1, we know the line number of the evidence, thus we just search the answer inside the line. For MS MACRO, we need to manually search the location, and if the answer shows up multiple times, data will be dropped. For some dataset, there are multiple acceptable answers, we will prefer to choose the answers with smaller length because we prefer our model to give brief answers. And, as you can refer, we filter out all yes/no questions.

Fix the answers: For answers cannot be found in the passage, we applied some fixing procedures before permanently drop it. We try to put some tolerance on cases, pronoun word, punctuations. For example, the passage might have "Panda is a bear native to south central China, …", and the answer might be "It is a bear native to south central China.". In this case, we will remove the "It" and the punctuation from the answer thus it can be considered as a range of passage.

By applying the steps above, we extract the datasets into valid passage-question-answer pairs. We keep the indexed corpus in a middle format for so both models can use them with the smallest effort.

4 Methods

4.1 End to End Memory Network

End to End Memory Network is proposed in 2015. It is a RNN based model. Basically. it has two parts: memory module and controller module. Memory module can get access to the passages multiple times, and this helps to aggregate more information of the given story for answering the questions. Controller module is to control the model focus on the parts which are related to the questions when reading the passage.

The structure of single layer End to End Memory Networks is shown in Figure [1]. The story is made up of several sentences, and each question is a single sentence. Firstly, the sentences and question are embedded with embedding matrices A, B and C (Each layer has their own embedding matrixes A and C). After soft-max and inner product, the corresponding weights for the sentences in the story are calculated, where the more related sentences get higher probabilities. O is the weighted sum of the sentences, representing the original story under consideration of the question. And $\hat{a}$ is the predicted answer.

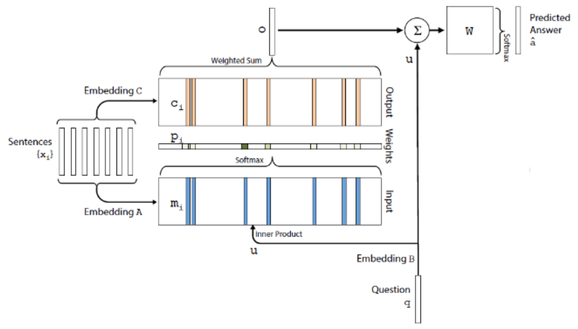

Fig. [1] The structure of single layer End-To-End Memory Networks

4.2 Position Encoding

In our experiment, we use a position matrix suggested in original paper to encode the word position information. To be specific, we have a weight matrix of size L×d (d is the dimension of embedding) and use this equation to generate the weights:$W_{kl} = (1 - l / L) - (k / d)(1 - 2l / L)$. In this rest of this sections, we assume all the sentences has been position encoded.

4.3 Layer details

Input Embedding: We use embedding matrices to transfer the text to vector spaces. For sentences $\{x_i\}$ in passages and the question $\vec{q}$, use embedding matrices A, B (both are d×V), to convert them in the memory space $\{m_i\}$. Then use soft-max to get the weight of each sentence: $p_i = softmax(u^T m_i)$.

Output Representation: Another embedding matrix C (d×V) is used to compute the story representation $\{c_i\}$. set $\vec{o} = \vec{u} + \sum \vec{c}_i * \vec{p}_i$ we get output of this layer. With the probability distribution of the last soft-max operation, we compute the weighted sum, which is the story representation when we consider the question.

Answer Prediction: The output representation o and question representation gather together, and pass the weight matrix with another soft-max to get the predicted answer. if we know the answer is a single word, we can just pick the one with highest probability as predicted answer.

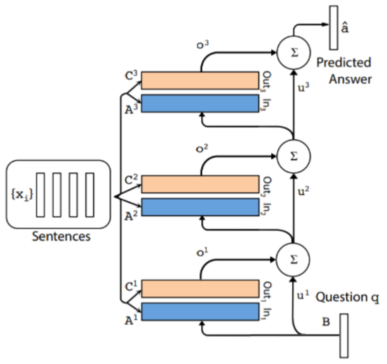

Fig. [2] The structure of 3-layer End-To-End Memory Networks

Figure [2] shows the structure of three-layer version of End-To-End Memory Networks. The output of the last layer is input of the next layer. Here we apply Adjacent method in End-To-End Memory Networks as weight tying method. Specifically, we set $A^{k+1} = C^k$. Also, we restrict $W^T = C^k$and $B = C^1$.

4.4 R-NET

R-NET is an end-to-end neural network model for machine comprehension and question answering. It is proposed by Microsoft Research Asia in 2015, and achieve good performance on Stanford Question Answering task. The model is made up of four parts: text encoding, question-passage matching, passage self-matching and answer prediction. The structure of the model is figure [3].

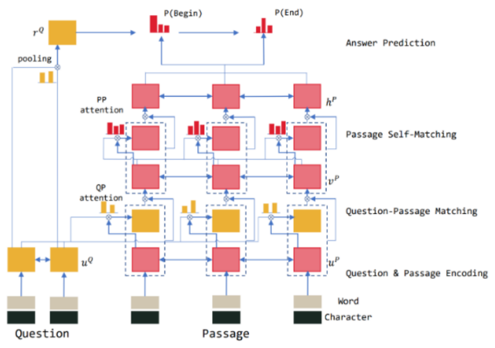

Fig. [3] The R-NET Structure

Encoder Layer: Word-level embedding and character-level embedding are both used in this part, for better dealing with OOV (out of vocabulary) problem. We use a bi-directional RNN to compute the question and passage representation:

$$u_t^Q = BiRNN_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$

$$u_t^P = BiRNN_P(u_{t-1}^P, [e_t^P, c_t^P])$$

Here $\{e_t^Q\}_{t=1}^m$ and $\{e_t^P\}_{t=1}^n$ are word-level embeddings, and $\{c_t^Q\}_{t=1}^m$, $\{c_t^P\}_{t=1}^n$are character-level embedding. m is the length of question and n is the length of the passage.

Question-Passage Matching Layer: A gated attention-based neural network is used to compute passage representation when considering of the question. The model proposed by Rocktaschel is used here. Specifically, two RNN is used for question and passage respectively, but initiate the second RNN with the last cell state of the previous RNN. In this task, we consider questions as the premise and passage the hypothesis. The output of this layer is $\{v_t^P\}_{t=1}^n$.

Passage Self-matching layer: This layer is similar to question-passage matching layer, but doing attention with passage itself. One drawback of RNN is that it only has a limited size of memory of context, and in long the sentence there is only a little information of the words far from the current state. To solve this problem, the passage is matched with itself to aggregate the whole information. The output of this layer is $\{h_t^P\}_{t=1}^n$.

Output Layer: A pointer network is used to generate the start and end position of the original passage. The hidden state is initiated by the attention-pooling over the question representation. Here the output is two pointers, start pointer and end pointer which give a sub-span of the passage as the predicted answer.

5 Experiments

5.1 End to End Memory Network

End to End Memory Network only works on single word answers (the original paper mentioned some methods to apply it to multiple word answers, but we don't think those are easy to approach). We applied it on bAbI dataset and SQuAD dataset. We didn't apply it to MS MACRO dataset because almost all the answers are not single word answers.

The results of bAbI tasks are shown in figure [4]. We use 100 epochs for training, and 0.01 as learning rate for SGD.

bAbI task	**Accuracy**
1 (one supporting fact)	0.995
2 (two supporting facts)	0.799
3 (three supporting facts)	0.566

Fig. [4] Results of bAbI tasks

To reduce overfitting, we add gradient noise [11]. It is obvious that with more facts to answer the questions, the accuracy is getting lower.

We notice that this model performs perfectly on bAbI dataset, it can archive very high train/test accuracy even for non-factoid questions logics. But in SQuAD data set, it doesn't give reasonable answers at all. We think this model is susceptible to vocabulary size. Which means without sufficient data, it might not be used in real-world question answering tasks.

5.2 R-NET

R-NET works very good on bAbI dataset task 1. Because of the char-level embedding, it can even give very good prediction on Hindi version (if we train it with bAbI dataset only). For SQuAD data set it also archived a good grade, which can be listed between 40 to 50 on SQuAD leaderboard. For MACRO dataset, the F1 score is more than 0.5, considering the MACRO dataset is real-world questions and answers with complex situations, we think that score is acceptable.

During the experiments of R-NET, we encountered 2 main problems:

Overfitting: The model can overfitting easily, even with appropriate dropout. We think this problem is mainly caused by lack of training data, which can be proved by an experiment that when using half the data, the model overfits much faster. At the same time, we need the attention size and layers numbers to be big enough to get a good training accuracy. To solve the problem, we think it's essential to get more training data. Luckily by applying the semi-supervised methods mentioned before, we can get theoretically unlimited data. But it requires some human work, which is time-consuming. We implemented an online algorithm of R-NET, thus our model can pick up new data when it's available.

Local optima: The second problem is local optima, which exists in both the models we use. So far, we don't have a perfect solution to cope with local optima, especially in R-NET. We think the technologies like Ensemble Methods might be a good way to solve this issue.

5.3 Final scores

Below are the test scores in the best trained model (R-NET single model)

	Exact Match	**F1**
bAbI (task 1)	0.84	0.92
bAbI (task 1, en & hn)	0.58	0.73
SQuAD	0.66	0.75
MACRO	0.44	0.53

Fig. [5] Results of final model

The precision and recall are defined as follows:

$$precision = \frac{count(same\ tokens)}{count(prediction\ tokens)}$$

$$recall = \frac{count(same\ tokens)}{count(standard\ answer\ tokens)}$$

5.4 Further Analyses

By manually analyze the answer generated by our model, we notice that this model has very good performance on the questions with short answers. For example, the questions like "Who did", "How many", "When did" can usually get a perfect answer. However, in questions like "What is" or "Why", our model prefers short ranges, thus usually generated a short answer which is a part of the full correct answer. Here is a short example:

Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation.

For the questions "What are the challenges of NLP?", our model might give "speech recognition" only. We think this problem might be cause by the biased dataset SQuAD and the way we preprocess the data. By applying some traditional NLP technologies like CFG might partially solve the problem.

By some experiments, we also noticed that the character-level does not contribute much is most of the OOV questions, but in OOV answers which contains numbers and time strings it gives good results. So, we think maybe its optional to have character-level embedding. Figuring out how to generate vectors for numbers and times might be a better approach.

6 Future Directions

6.1 Ensemble Methods

For all the experiments, we use single model. However, both models have local optima issues. Thus, we think ensemble methods should be able to contribute to both models greatly.

AdaBoost seems to be a good choice, but might need a long time or good computing resource to train the model.

Alternatively, these models can work together with a yes/no QA models. All the questions answer pairs can be converted to a yes/no question. For example, "When did Andrew come home?" and "5 PM" is equivalent to "Did Andrew come home at 5 PM". There are many models which perform very good on yes/no QA tasks. We can use our models to generate candidate and use those high accuracy yes/no QA models to give final decision of the answer.

6.2 Word2vec for End to End Memory Network

One biggest limitation of End to End Memory Network is that its sensitive to vocabulary size. We think bringing Word2vec to this model might be a good way to solve the problem. So instead of using simple position encoding, we could apply some encoder layer like the encoder layer in R-NET. But there are many details need to figure out. For example, currently the matrix A and C have size (d×V), apparently, we need to put a constraint on V, but should we change V to the dimension of the word vectors? How we generate the word from the final output? There are too many questions and options, answering those questions will need many experiments.

6.3 Use better Work2vec for R-NET

It seems to be relatively hard to bring big improvements to R-NET. We tried many small changes during the implementation (for example, use LSTM instead of GRU, or use larger attention size). However, due to the local optima issue, we cannot know tell how much it proves.

But one thing apparently can help is to use a better Work2vec. GloVe is designed for general purpose. A Work2vec designed for question answering purpose might greatly improve the performance of some tasks. For example, many numbers are missing in GloVe. We can extend GloVe with one more dimension, and the value will be the value of the number. For embedding, we check if this is a number, and if it's a number, we dynamically create a new vector for this number. We can do the same thing for the time strings. We can even insert sequence words (time words in End to End Memory Network) into sentences thus R-NET might able to answer some sequence related simple non-factoid questions.

6.4 Phrase Structure and Language Models

As we mentioned above, our models sometimes give short answers which lays in the full correct answer. One possible way is to use traditional NLP technologies to improve the answer. We can combine the generated result with the phrase structure of the passage, we can also generate a list of possible phrases which contains the predicted answer. And then we use language models to check which answer makes more sense.

For example, for questions like "What are the challenges of NLP?", we might have candidates "speech recognition", "speech recognition, natural language understanding, and natural language generation." and "involve speech recognition, natural language understanding, and natural language generation.". Then we can compare the possibility of "The challenges of NLP are {answer}", and because of the word "challenges" and "are", the languages model will probably say the second answer has the biggest possibility.

7 Acknowledgments

We would like to thank Min Sang Kim and Vincent Huang, both have some previous work in R-NET which greatly saved our time. Also, they actively answered our questions and have shared their own analysis and suggestions about the R-NET which are great references for our project.

REFERENCES

[1] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman,and Phil Blunsom. 2015. Teaching machines to read and comprehend. In Proc. of NIPS, pages 1684–1692.

[2] Sukhbaatar, Sainbayar, Jason Weston, and Rob Fergus. "End-to-end memory networks." Advances in neural information processing systems. 2015.

[3] Danqi Chen and Jason Bolton and Christopher D. 2016. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task.

[4] R-NET: Machine Reading Comprehension with Self-matching Networks. https://www.microsoft.com/en-us/research/publication/mrc/

[5] The bAbi Project. https://research.fb.com/downloads/babi/

[6] Yoav Goldberg, Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method.

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

[8] Yoav Freund Robert E. Schapire, "A Short Introduction to Boosting" http://www.site.uottawa.ca/~stan/csi5387/boost-tut-ppr.pdf

[9] Wang S, Jiang J. Machine comprehension using match-lstm and answer pointer[J]. arXiv preprint arXiv:1608.07905, 2016.

[10] Rocktäschel T, Grefenstette E, Hermann K M, et al. Reasoning about entailment with neural attention[J]. arXiv preprint arXiv:1509.06664, 2015. MLA

[11] Neelakantan A, Vilnis L, Le Q V, et al. Adding gradient noise improves learning for very deep networks[J]. arXiv preprint arXiv:1511.06807, 2015. MLA

[12] Nguyen T, Rosenberg M, Song X, et al. Ms marco: A human generated machine reading comprehension dataset[J]. arXiv preprint arXiv:1611.09268, 2016.

[13] Kumar A, Irsoy O, Ondruska P, et al. Ask me anything: Dynamic memory networks for natural language processing[C]//International Conference on Machine Learning. 2016: 1378-1387.

[14] Jurafsky D, Martin J H. Speech and language processing[M]. London: Pearson, 2014.

[15] Wang S, Jiang J. Learning natural language inference with LSTM[J]. arXiv preprint arXiv:1512.08849, 2015.

[16] https://github.com/carpedm20/MemN2N-tensorflow

[17] https://github.com/minsangkim142/R-net

[18] https://www.tensorflow.org/tutorials/