**Short Passages Reading Comprehension and Question Answering**

Project Progress Report

|  |  |  |  |
| --- | --- | --- | --- |
|  | Yuan Yang Northeastern University [yang.yuan1@husky.neu.edu](mailto:yang.yuan1@husky.neu.edu) | Zhe Hu Northeastern University [hu.zhe@husky.neu.edu](mailto:hu.zhe@husky.neu.edu) |  |

1. Changes

No change has been made for task, datasets and evaluation methods. For models, we encountered some limitations on the model we picked (End-To-End Memory Networks). We will discuss the details of the limitations in the section 4 and 5. We are trying to solve the problem by improving the model. Meanwhile, we want to try some other models such as R-NET or Match-LSTM in the future.

1. Data Preprocessing

We have data from multiple datasets. All of three datasets are well preprocessed, we don’t need to do extra cleaning and selection work. We just need to convert each dataset into some standard format so that they can be picked up by our model. We have finished the data preprocess work for all three datasets.

Our data contains 3 parts: P (passage), (question), (answer). P is a M×L matrix where M is the number of sentences in the story, and L is the max length of all sentences; is a 1×L vector; is a 1×V vector where V is vocabulary size.

To begin with, we designed a standard data format. Assuming we have N samples, the final passages should be a N×M×(L+1) matrix, the questions should be a N×(L+1) matrix, the answer is a N×(V+M) matrix. All words must be converted to word indexes. We added “time words” at the beginning of all sentences in the passage. Time word are some words which doesn’t exists in vocabulary (for example “time\_1”, “time\_2”), but they helped to keep sequence information. In the rest of the report, we will use V to replace V+M, and use L to replace L+1.

**bAbi dataset**

Process bAbi dataset is relatively easy. All 1000 sample passages, questions and answers contain only 20 words. Each passage contains exactly 10 sentences and the max length of a sentence is 6 (including question sentences). All the answers are single word answers. We just load the data and use index information to recognize passages questions and answers, and then tokenize the words and remove the terminators like question marks. Then we index the words and converted into the standard format we defined.

For each question type, we load all the passages into a 1000**×**10**×**7 matrix, all the questions into a 1000**×**7 matrix and all the answers into a 1000**×**30 matrix. The following figures show the first sample in the training set and its corresponding matrix form (for the first sample, we only know the first two sentences, so only the first two dimensions are nonzero).

1 Mary moved to the bathroom.

2 John went to the hallway.

3 Where is Mary? bathroom 1

4 Daniel went back to the hallway.

5 Sandra moved to the garden.

6 Where is Daniel? hallway 4

7 John moved to the office.

8 Sandra journeyed to the bathroom.

9 Where is Daniel? hallway 4

10 Mary moved to the hallway.

11 Daniel travelled to the office.

12 Where is Daniel? office 11

13 John went back to the garden.

14 John moved to the bedroom.

15 Where is Sandra? bathroom 8

[[21 8 17 16 15 6 0]

[20 11 9 16 15 2 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]

[ 0 0 0 0 0 0 0]]

Fig. [1]: The original data and passage matrix

**Stanford Question Answering Dataset (SQuAD)**

SQuAD data is in JSON format. Different from bAbi dataset, SQuAD dataset provides more than one possible answers for each question. So, for each unique answer we duplicate the passage and question and treat them as a separated train sample. We also need to index the word so that the data can be converted to standard format. For SQuAD training data, M=30, L=400, V=28000, N=88000. I this dataset V is sufficiently large.

**MS MARCO Dataset**

MS MACRO Dataset is also in JSON format, and it is the largest dataset among the three. Different from other datasets, MS MACRO dataset provides more than one passage for each question and at least one of the passages contains the answer (and the passages contain the answer are labeled). MS MACRO dataset also provides more than one answers for each question. For now, we choose to combine the passages into a single big passage, and duplicated the passage and question for different answers. For MS MARCO training data, M=200, L=1100, V=950000, N=90000. All dimensions are very large in MS MARCO data set.

1. Model

**Why End-To-End Memory Networks**

We did some searching work on existing models which can be used to solve QA tasks. Most of the models are either very complex or limited on some certain task types. Comparing with other models, End-To-End Memory Networks is relatively general and simple and need less supervision during training, which convinced us to use this model as first step. It is a novel recurrent neural network (RNN) architecture, based on Memory Networks structure[2][7].

This model does not have strong requirement on the question types. The bAbi dataset contains 20 different questions types such as “Where is [entity]?” or “How many objects is [entity] carrying?”. The questions in other two datasets are much more general, such as “Which NFL team represented the AFC at Super Bowl 50?”. Flexibility of answering different kinds of questions is obligatory in our task. This model is also much simpler comparing with other QA models. Considering we don’t have many related research experiences, so we think we should start from a simple model which we can understand and improve.

Here is the three-layer End-To-End Memory Networks [2] we use in this project.

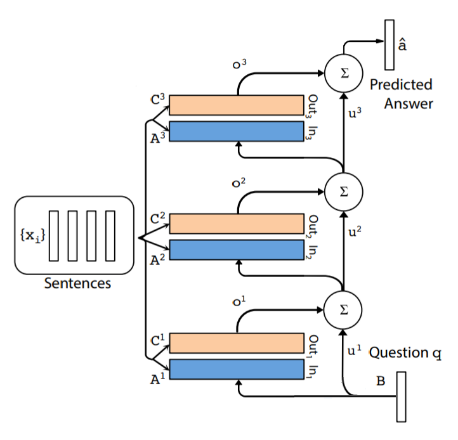


Fig. [2] The structure of 3-layer End-To-End Memory Networks

**Position Encoding**

In our experiment, we use a position matrix 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: . In this rest of this sections, we assume all the sentences has been position encoded.

**Layer details**

Each layer takes (a vector with size V) and (one passage) as inputs. For each sentence in , we embed it with a matrix A (d×V) and we get a memory . After that we compute the inner product between and and use softmax to get the probabilities . We embed again with a matrix C (d×V) and we get , set we get output of this layer, we can pass it to the next layer if this is not the final layer. Each layer has their own embedding matrixes A and C.

For the question , embed it with embedding matrix B (d×V) and we get . Which can be used as the input of first layer. When we get the output of last layer, if we know the answer is single word answer we just pick the one with highest probability. We haven’t implemented the case that the answer might contain multiple words.

Also, we used adjacent weight typing method in out model as described in [2]. Specifically, the output embedding for one layer is the input embedding for the one above. We also constrain (a) the answer prediction matrix to be the same as the final output embedding, and (b) the question embedding to match the input embedding of the first layer.

1. Results

**Current Achievement**

We have successfully trained and tested the model with bAbi dataset. The result varies on question types. For now, we can reach 99.0% test accuracy on some tasks while can only have 12.5% test accuracy on 1 task (path finding). The average accuracy of 20 tasks is around 85% (all tasks have same number of train and test samples). To achieve this accuracy, we use three layers and set the embedding dimension d to V (20).

**RAM Usage and Unknown Word Issue**

We tried to run this model on SQuAD and MS MARCO datasets. However, it was extremely slow and didn’t seems to be able to finish. We noticed that the training program ate RAM very fast and quickly tricked page swapping and became slow. Due to this large memory usage issue, we haven’t able to test this model on other datasets.

We noticed that the vocabulary size V in SQuAD and MS MARCO are very large, which directly leads to the RAM usage issue. It seems to be necessary to remove low frequency works to unknown tokens. However, this model requires the answer is one of the known words, and it very common that a question/answer contains a rare word. For example: “what is tesgüino?”.

To reduce V, one possible solution is to figure out the words which can be abstracted, and we replace them with internal words. For example, “John travelled to the hallway. Where is John?” can be changes to “<OBJ1> travelled to the <OBJ2>. Where is <OBJ1>?”. This can both control the V size and solve the unknown word issue.

1. Next Steps

**Multiple Word Answers and F1**

So far, we only implemented answering questions with single word. If the answer might contain multiple words, we can pick words with much higher probabilities than others and try to generate a meaningful phrase. We are trying to find way to make sure we won’t pick up too much words or too less.

Because the output of our model doesn’t contain sequence information, it’s unlikely to generate the exact same answer if it contains multiple words. In that case F1 can give a more reasonable evaluation about the generated result. Slightly different from what we learned in class, the precision and recall is defined as follows:

For single word answers, F1 will has the same value with EM (Exact Match).

**Robustness**

This model is sensitive to 3 numbers: passage max sentences length M, sentence max word length L and vocabulary size V. In that case, apply this model in SQuAD and MS MARCO dataset is becoming a very big challenge. We are still working on that but it doesn’t look promising. As we discussed in section 4, we’ve already have some primitive ideas to control the V. In addition to that, we could also try to reduce the M (number of sentences in a word). We might able to use some windowing technology to control the passage size. An alternative solution could be use other models to find the relevant sentences first and then apply this model on most relevant sentences only.

**Embedding**

What we have done so far is use position encoding to do sentence embedding. We also want to try some other advanced embedding methods like word2vec or doc2vec to make the embedded content more meaningful and improve the accuracy of our model.

**Other Models**

We are trying to improve this model to solve all the issues. We will not directly switch to other models unless we are certain that some critical problems are not solvable. But to reduce the risks, we have started investigating other models which don’t have these issues, such as R-NET and Match-LTSM. Right now, we are trying get ideas from other models.

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. |