CS224N Final Project: QA

Te-Lin Wu
tlein@stanford.edu

De-An Huang
dahuang@stanford.edu

Abstract

We extend existing question answering (QA) system to deal with words that are unseen or do not have enough examples during training. Instead of learning word embedding from scratch for a specific QA dataset, we decompose the embedding into two components: The first captures the general semantic meaning of a word and can be trained using readily available large corpuses. The second component reweights the general embedding so that it is optimized for our QA tasks. The advantage is twofold. First, the number of parameters to learn using the QA dataset does not grow linearly with the size of the vocabulary. Second, the system can be easily generalized to unseen words by borrowing the knowledge from outside corpuses.

1 Introduction

Teaching machines to read and reason on top of natural language documents remains an elusive challenge (Hermann et al., 2015). To measure the progress towards this goal, (Weston et al., 2015) recently proposed a set of proxy tasks (bAbI tasks)[1] that evaluate reading comprehension via question answering. While the tasks are well-defined and based on a unified underlying simulation of a physical world, the tasks remain challenging as it requires the system to reason on top of several supporting facts that are *automatically* selected. For example, given the story:

John is in the playground.
Bob is in the office.
John picked up the football.

[1] https://research.facebook.com/researchers/1543934539189348

To answer the question *Where is the football?*, the system need to first infer that the football is at the same location as John, and answer John's location from the story.

An important aspect to answer such questions is to memorize the story and retrieve the relevant fact given a question or query. In principle this is possibly achievable by a language modeler such as a recurrent neural network (Hochreiter and Schmidhuber, 1997), as these models are able to encode a stream of words and use it to generate the desired output. However, its memory is typically too small to deal with longer story. To address this problem, (Weston et al., 2014) proposed *memory networks* with explicit read and write operations and achieve significant improvement on the synthetic bAbI tasks (Weston et al., 2015). This model is further extended to be trained in an end-to-end fashion (Sukhbaatar et al., 2015).

In this project, we are interested in bringing the success of memory networks beyond the limited vocabulary of the bAbI tasks (about 150 words), and possibly to a wider range of questions. This is in line with recent works (Hermann et al., 2015; Hill et al., 2015) which apply memory networks or neural network models to larger QA datasets beyond synthetic data.

Instead of directly training an existing model on newer and larger datasets, we propose to extend the word embedding layers of memory networks so that models trained on synthetic dataset with limited vocabulary can be easily extended to story and questions composed of unseen words during training. More specifically, we decompose the embedding layers of memory networks into a semantic embedding using GloVe (Pennington et al., 2014) and another linear projection layer that adapt the GloVe word representation to the QA tasks.

As only this linear projection from GloVe to memory networks is learned during our training, the number of parameters to learn does not grow linearly with the size of the vocabulary (already handled by GloVe), and our model can be easily applied to words that are unseen at training time given its GloVe word representation.

2 Approach

The setting of our question answering problem is as follows: First, the system is given a *story* $X = \{x_i\}$ composed of a set of sentences. The story contains the required information to answer a *question* q. For each question q, the answer will be a single word a in the vocabulary of size V.

2.1 Memory Networks

In this section, we briefly summarize the memory networks (Weston et al., 2014; Sukhbaatar et al., 2015).

Input memory. The given story $\{x_i\}$ is first stored in the input memory as memory vectors $\{m_i\}$. This is achieved using an embedding matrix A of size $d \times V$ where d is the dimension of the memory vector. For the simplest bag-of-words scenario, the memory vector is computed by

$$m_i = \sum_j A x_{ij}, \tag{1}$$

where x_{ij} is the j-th word of the i-th sentence in the story. Given a question q, it is first also embedded using another matrix B into u.

The relevance p_i of each memory vector m_i given the question embedding u can be computed by

$$p_i = \text{Softmax}(u^T m_i) \tag{2}$$

comparing the memry vector against the question embedding.

Output memory. Instead of using the weighted memory vector $\sum_i p_i m_i$ directly as output representation, the memory networks embeds the story using another embedding matrix C to produce the output. Again, the output vector c_i is computed by $c_i = \sum_j C x_{ij}$ in the simplest bag-of-words case. The final response from the memory is then given as $o = \sum_i p_i c_i$. The idea is that the embedding C should capture important information in the story for output, while the embedding A should extract informatoin for comparing the story with the query.

Final prediction. The final prediction $\hat{a}$ is made by a fully connected softmax layer

$$\hat{a} = \text{Softmax}(W^T(o + u)) \tag{3}$$

where W is the weight matrix of size $d \times V$.

Such memory layer could be further stacked and combined with temporal encoding instead of the BoW representation to improve performance. Please refer to (Sukhbaatar et al., 2015) for details. Our implementation is based on their full model, which is available online.[2]

2.2 Incorporating External Corpus

The embeddings A, B, and C and weight W previously introduced are all of size $d \times V$, where V is the size of the vocabulary. Effectively this means that, given a new QA dataset, the system needs to learn new embeddings and weight for each word in the vocabulary from scratch. Without enough or even any instance of a word in the training data, the quality of the learned embeddings is limited. While the synthetic bAbI dataset can potentially generate infinite example, the vocabulary size is still limited (about 150 words).

To tackle real world question answering, a promising direction is to collect QA datasets of much larger scale (Hermann et al., 2015; Hill et al., 2015). In this project, we pursue a different strategy that tries to incorporate knowledge acquired from other readily available datasets. Take the embedding A as example, we can decompose the $d \times V$ matrix into

$$A = P_A E, \tag{4}$$

where $E \in \mathcal{R}^{e \times V}$ is the semantic word embedding that captures the general meaning of the word, and $P_A \in \mathcal{R}^{d \times e}$ is a linear project that adapts the general semantic embedding to specific purpose of A.

The embedding E can be trained using word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) using large online corpus. Specifically, we use GloVe trained on Wikipedia 2014 + Gigaword 5, and the dimension $e = 100$.

The project P_A is of a fix size $d \times e$ independent of the size of the vocabulary. And can be robustly estimated with more training data in the QA dataset. This significantly reduces the number of parameters to learn. Furthermore, As only P_A

[2] `https://github.com/facebook/MemNN`

is required to be learned during training, the embedding $A = P_A E$ can be extended to arbitrary vocabulary given the embedding E. The embeddings B, C and weight W can also be decomposed simiarly.

Optimization. We derive backpropagation of P_A so that it can be optimized together with the rest of the network. In this case, we can assume that the gradient for A is given. Let $\mathcal{L}$ be the loss, then our task is to derive $\frac{\partial L}{\partial P_A}$ given $\frac{\partial L}{\partial A}$. For clearer presentation, we use P as P_A in the derivation. First we have

$$\frac{\partial \mathcal{L}}{\partial P_{ij}} = \sum_m \frac{\partial \mathcal{L}}{\partial A_{im}} \frac{\partial A_{im}}{\partial P_{ij}} \tag{5}$$

because P_{ij} will only contribute to A_{im}. By definition we have

$$A_{im} = \sum_j P_{ij} E_{jm} \tag{6}$$

$$\Rightarrow \frac{\partial A_{im}}{\partial P_{ij}} = E_{jm} \tag{7}$$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial P_{ij}} = \sum_m \frac{\partial \mathcal{L}}{\partial A_{im}} E_{jm} \tag{8}$$

$$\Rightarrow \frac{\partial \mathcal{L}}{\partial P} = \frac{\partial \mathcal{L}}{\partial A} E^T. \tag{9}$$

The gradient for P_B, P_C and P_W can also be derived similarly.

3 Experiments

3.1 bAbI Tasks Accuracies

First we compare the performance of our embedding to the original end-to-end memory networks (Sukhbaatar et al., 2015). The error rates are shown in Table 1. MemN2N (Sukhbaatar et al., 2015) is the original end-to-end memory networks with three layers of memory. Here 3 hops means three layers of memory and 1 hop means 1 layer of memory.

In general incorporating the semantic embedding trained on outside dataset makes performance worse. This is understandable as some words that should be discriminated are forced to have similar word vector in our model. For example, an important cue for question answering system to pick the correct supporting facts is to recognize the subject in the question. However, most of the subjects in the dataset, such as “Jeff” and “John”, have similar GloVe representation, and therefore it is harder for our system to reason on top on sentences involving several subjects.

Task	MemN2N 1hop	Ours 1hop	MemN2N 3 hops	Ours 3 hops
1: 1 supporting fact	0.8	0.5	0	0.2
2: 2 supporting facts	62	82.5	19.7	69.6
3: 3 supporting facts	76.9	78.1	29.2	70.2
4: 2 argument relations	22.8	25.9	10.5	29.6
5: 3 argument relations	11	15.8	14.4	16.7
6: yes/no questions	7.2	36.9	2.1	36.2
7: counting	15.9	17.4	16.8	22.5
8: lists/sets	13.2	12.4	10.9	16.4
9: simple negation	5.1	26.4	2.4	27.1
10: indefinite knowledge	10.6	41.4	6	40.4
11: basic coreference	8.4	19.9	1.5	14
12: conjunction	0.4	1.2	0	0.8
13: compound coreference	6.3	5.8	0.5	6.3
14: time reasoning	36.9	44.9	5.1	33.6
15: basic deduction	46.4	71.5	8.3	59.9
16: basic induction	47.4	51.2	0.8	51.4
17: positional reasoning	44.4	52	44.2	48.9
18: size reasoning	9.6	37.6	8.7	50.1
19: path finding	90.7	91.5	86.9	89.6
20: agent’s motiation	0	0.5	0.2	0.1
mean error	25.8	35.7	13.4	34.2

Table 1: Error rates for different models. Lower is better.

A closer inspection shows that our 3 hops system is performing well for tasks that only involve one supporting fact, such as “1 supporting fact” and “conjunction”. This means that it is harder for our system to have different behavior in different layer of the memory. This is verified by the fact that the learned input embedding of our second and third memory layers are much more correlated (correlation coefficients 0.4503) than that of MemN2N (correlation coefficients 0.1280).

It should be noted that our 3 hops performance with the embedding is indeed similar to our 1 hop performance, which is again similar to 1 hop MemN2N except a few tasks.

3.2 Handling Unseen Words in Story

3.2.1 Hypothesis

We then conducted the core experiments of our investigation, changing some of the vocabulary in the testing dataset such that those changed words have never occurred in the training corpus and they share high similarities with the original words in terms of human common sense. In this sense, those modified words have never been seen by the End-to-End Memory network, as a result, we expected that our improved model would perform better than the original model using only one-hot representation because our model tends to preserve the information of similar words using Glove. Take a look at the one supporting fact question answering task as an example as shown below:

```
Mary journeyed to the restroom.
Daniel went back to the restroom.
John moved to the bedroom.
Where is Mary? bathroom
```

We changed the word "bathroom" in the supporting facts to "restroom" in the testing task, such that in common sense these two words are actually meaning the same thing, and of course, having high word similarity in terms of their word vectors generated by Glove. The original model only using one-hot representation would fail to identify this whole new word, resulting in worse performance by a significant drop compared to the original non-changed testing dataset. And to a human, we can logically refer the word "restroom" to "bathroom" due to their identical meaning, so we can easily verify that the answer in this case is correct. Our proposed model having knowledge base from the entire wikipedia as the word embedding is capable of doing this type of task and still perform well. The word modification experiments that we conducted and their corresponding numerical results are shown in Table 2.

3.2.2 Experiments

Since the vocabulary for each task is actually relatively small, we could only choose from very limited words to test our method and verify our hypothesis. After some simple statistics, we picked words that were fundamental and essential for the entire question answering task, and replaced them with very similar or even identical words. Observing the results, it is easy to notice that, comparing to the results from Table 1, the performance of MemN2N, the original model, have been damaged significantly when some essential words have been modified. We mainly picked the tasks that even using the orginal training dataset, our proposed model can still perform relatively well, even though no better than the original MemN2N, namely, qa1, qa5, qa12, qa13, qa20. Our model out performed the original one in task qa1, qa12, qa13, and qa20. If the words are used as some of the answers, like bathroom or office, the performance varied a lot between MemN2N and our model, since they're exactly the vital information for in those supporting facts. Those words are usually noun or noun phrase. Interestingly, even verbs can affect the overall outcome, in qa20 we

Task	word	new word	Mem2N2N	Ours
qa 1	bathroom	restroom	12.5	1.21
qa 1	bathroom	toilet	14.11	1.01
qa 1	office	workplace	15.93	6.45
qa 12	bathroom	restroom	17.74	2.22
qa 12	office	workplace	10.81	6.35
qa 13	bathroom	restroom	12.60	7.56
qa 13	office	workplace	15.83	12.5
qa 20	hungry	starving	13.21	7.06
qa 3	bathroom	restroom	54.53	76.41
qa 5	bathroom	restroom	13.01	17.14
qa 5	office	workplace	11.59	17.64
qa 2	bathroom	restroom	12.5	82.26

Table 2: Error rates for vocabulary modification.

tested if we change hungry to starving, meaning very much the same situation of a character, and our model outperformed twice as good as the original one. For task qa2, qa3, and qa5, our model failed to beat the MemN2N, since multiple supporting facts require using longer term memory, and the exact words are crucial to the overall performance. But if the vocabulary size of the testing dataset is actually as large as our knowledge base, wikipedia or some equivalent base full of plenty information, the very limited one-hot representation of MemN2N will fail to generalize the result. We viewed this flexibility for word embedding with different testing task very vital since this will actualize the real AI-Complete Question Answering system.

3.3 Extension to

Our model can actually extend the output matrix W to output vector of any size, as long as it doesn't exceed the total size of the vocabulary we have from our corpus when training the Glove. In such sense, we can observe what are the words that the top k scores of the output vector represent. We extracted a single self-contained story from the testing dataset, and observe what will be the resulting output vector for the final question answering task for such chosen story. We conducted the experiments on qa1, qa12, and qa13, since they show the largest difference between the performance of MemN2N and our model in the previous section.
The following blocks show the results corresponding to qa1, qa12, ans qa13, respectively. In the first block, which extracted a qa task from qa1 testing dataset, with the answer be bathroom in the particular example. The most accurate answer is denoted with color blue, the nine other words in color pink is the top 2 to 10 candidate words in our output vector W, and it's interesting to notice

that those words would share similarity with the answer itself. There are words such as restroom, lavatory, and toilet. In the second and third blocks, the results are as well showing the word with highest score as the answer, and the following top 2 to 10 words as possible candidate answers. One thing worth paying attention is that these minor answers were never occured in the training dataset. In other words, the memory network have never seen them, so it's impossible for the original model only using MemN2N to generate such results. Our model is capable of dealing with this task and can be fully generalized to any size of output vector. This gives the QA system with much more flexibility and enables possible usages that the training are omitted with some certain information, making the entire system more AI-Complete and even smarter.

The following are the mentioned experiments, blue colored words are the exact answers and magenta colored words are other top2 to 10 candidate words.

Mary journeyed to the bathroom.
Daniel went back to the bathroom.
John moved to the bedroom.
Where is Mary? bathroom
restroom, lavatory, toilet, bathrooms
cubicle, ensuite, bathtub, cubicles, workroom

QA1

Mary and Sandra went to the bathroom.
Daniel and John journeyed to the hallway.
Where is Mary? bathroom
Mary and John moved to the bedroom.
John and Sandra travelled to the office.
Where is John? office
offices, sub-post, residence, pmo, bureau
municipal, headquarters, deputy, midtown

QA12

Sandra and Daniel journeyed to the garden.
Then they went to the bedroom.
Where is Sandra? bedroom
offices, sub-post, residence, pmo, bureau
municipal, headquarters, deputy, midtown

QA13

References

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.