Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Question Answering with Recurrent Neural Networks

Seminar Paper

Neural Networks

Irina Barykina
6barykin@informatik.uni-hamburg.de

13.07.2017

Abstract

Memory-augmented models show excellent performance in the area of Question Answering due to the ability to keep and address a sufficient amount of facts in the memory bank. As soon as memory reads have a discrete nature, those models have to address memory as a whole and, consequently, exploit "attention mechanisms" to emphasize relevant facts. Such "attention mechanisms" depend on the semantic similarities between questions and facts in the memory, thereby, the representation of sentences plays a distinctive role. In this work, we investigated how End-to-End Memory Network behaves with three different approaches to the sentence encoding: namely, Bag-of-Words, position and LSTM encoding.

Contents

1 Introduction

Question Answering (QA) is the discipline in Natural Language Processing that aims to build systems able to either find in the text statements relevant to the question or provide an exact answer, expressed in a natural language. In this work, we address a more narrowed open-domain QA problem, that does not require domain-specific auxiliary knowledge and generates an answer using only the provided information. Systems with open-domain QA have found application in different areas like information retrieval or dialog systems.

Among neural network models, recurrent models are usually chosen to solve the QA problem, as they are able to handle sequential information. For instance, as shown by Wang and Nyberg [7] in their thorough assessment of different long short-term memory (LSTM) models, this type of recurrent neural networks (RNN) can achieve significant performance in an answer sentence selection task. However, despite this success, RNNs usually lack ability to use a long term memory component and as a result lose precision of the known facts by having to compress them in the small space of the hidden states.

Recently, the memory-augmented models of RNNs emerged to cope with the issue of keeping and addressing a large amounts of contextual information. Weston et al.[9] introduced the strongly supervised Memory Network with a potentially large long-term memory bank. The problem of this model is a discrete nature of its memory access that makes the end-to-end training impossible. So later this model was modified to the continuous form by Sukhbaatar et al.[5], who suggested avoiding discrete addressing of memory and instead using it as a whole. The relevance of the facts in this case is determined by the "attention mechanism". The similar idea was proposed by Graves et al.[1] for, so called, Neural Turing Machines. Their model, however, has read and write access to the memory bank.

One of the important design choices in the memory-augmented neural networks is a representation of the data. Similarly to the embedding of words, the embedding of the sentences plays a major role, as it reduces the dimensionality of the data and enables analysis of semantic proximity for the sentences. The latter is essential for scoring of the match between sentences, which is the base for attention mechanisms. Besides the naive approaches like Bag-of-Words or position encoding, the more sophisticated approaches to the sentence embedding were introduced recently. For instance, Doc2Vec [4] or Skip-of-Thought [3] models. However, not all of them could be incorporated easily to another model.

In this work, we are aiming to investigate different approaches to the vectorization of the sentences and how they could influence the performance of the models with a memory bank, in particular, continuous Memory Networks. We begin with a brief overview of the general architecture of Memory Networks and their differentiable analog End-to-end Memory Network in Section 2. In Section 3, we discuss different approaches to the sentence encoding: namely, bag-of-words, position encoding and encoding with LSTM. Section 4 provides general information about the bAbI dataset, used in our experiments. In Sections 5 and 6, we describe the details of experiments and evaluate models with the mentioned sentence encodings.

2 Background Information

2.1 Memory Networks

Memory Networks were introduced by Weston et al.[9] and the main advantage of this architecture is the ability to keep large amount of data in the memory component. The general architecture of Memory Networks consists of 4 different parts: I, G, O and R, where each of them could be either learned or predefined. I is an input feature map, that converts input data to the internal representation. For example, it might be conversion of words from one-hot to dense representation. G is a generalization component, which updates memory with newly arrived facts. It might be a simple write operation to the next free memory slot or an update of the whole memory. O is an output feature map, which produces output, given the current memory state and the input. Typically, this component determines and retrieves relevant memories. Finally, R is a response component, which converts output to the expected format, for example, a textual response. It might be an RNN producing the actual response, given relevant memories.

Thus, in the context of QA, the memory network has following workflow:

1. Encode input facts and a question to the internal representation $I(x)$.
2. Given the encoded fact $I(x)$, update a memory slot $m_i = G(m_i, I(x), m)$.
3. Compute the output o, such that, $o = O(I(q), m)$, where q denotes the question. O could be used several times to retrieve a sufficient number of supporting facts (so-called memory hops, denoted below as k). Given a score function s_o, matching relevance between pair of sentences x and m_i:

$$o_1 = O(I(q), m) = \arg\max_{i=1,..,N} s_o(q, m_i)$$

$$o_k = O(I(q), m, [o_{k-1}, .., o_1]) = \arg\max_{i=1,..,N} s_o(q, m_i, [m_{o_{k-1}}, .., m_{o_1}]), k > 1$$

4. Finally, decode the output to the expected representation. For example,

$$r = R(o) = \arg\max_{w \in W} s_r([q, m_{o_1}, m_{o_2}], w),$$

where W is the dictionary and s_r is a function scoring the match between memories and words from the dictionary.

The scoring functions might be as simple as a dot product between vectors. The training process requires strong supervision (supporting facts along with expected answers) due to the non-differentiable form of the memory component.

2.2 End-2-End Memory Networks

The very general model, proposed by Weston et al.[9] was later altered by Sukhbaatar et al.[5] to the continuous, differentiable form. This form could be trained end-to-end and requires only soft supervision, which is the more appropriate approach for

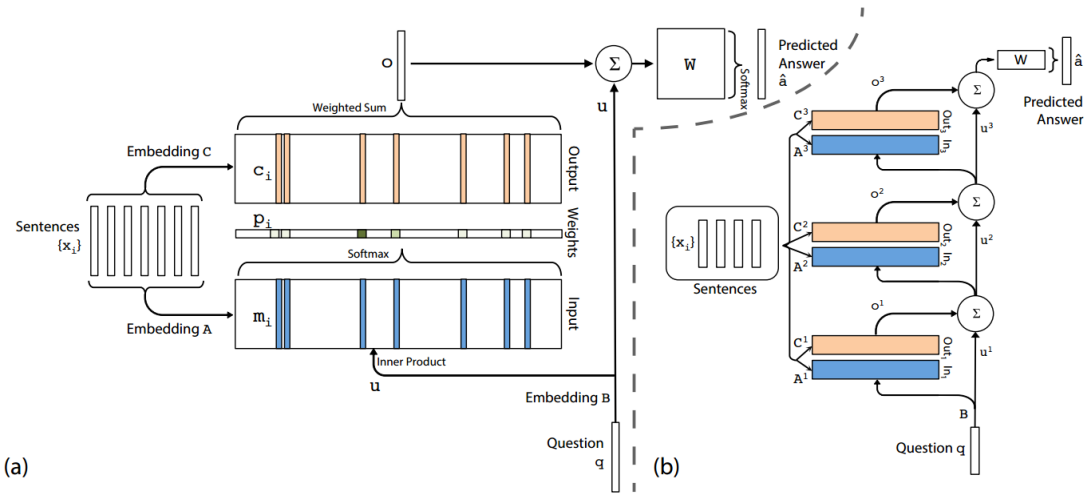

Figure 1: A single layer (a) and three layer (b) version of End-2-End Memory Network. Figure is adapted from [5].

the real-word tasks. The workflow of this End-to-end Memory Network is following (Figure 1):

1. Convert input statements $\{x_i\}$ to dense representations $\{m_i\}$, using an embedding matrix A. Also, convert a question q to u, using another embedding matrix B. Both matrices A and B here have dimension $d \times V$.
2. Compute the matches between u and $\{m_i\}$ as $p_i = softmax(u^T \cdot m_i)$.
3. Compute output like $o = \sum_i p_i \cdot c_i$, where c_i is another dense representation of x_i, computed with an embedding matrix C.
4. Generate the final prediction $a = softmax(W(o + u))$, where $W : V \times d$.

This model could be layered, so that the input to the next layer is the output from the previous one: $u_{k+1} = u_k + o_k$. Layers could share their parameter matricies A, C, B or use different ones. In the former case, the model forms the traditional RNN.

3 Sentence encoding

The story sentences in Memory Network are represented in the vectorized form. Sukhbaatar et al.[5] suggested two different methods of the sentence encoding:

- Bag-of-Words (BoW): the classic bag-of-words approach, introduced by Harris [2], applied in the embedded space. Every sentence is represented as a sum of constituent words in the dense form: $s = \sum_j E * w_{ij}$, where E is an embedding matrix, w is a word encoded as one-hot vector and s is a vectorized representation of a sentence.

- Position Encoding: the encoding of the sentence that takes into account the order of words. So that $s = \sum_j l_j \cdot Ex_j$, where E is an embedding matrix, l_j is a column vector with elements $l_{kj} = (1-j/J)-(k/d)(1-2j/J)$, J is the number of words in the sentence and d is the dimension of the embedding space. Thus, a sentence is represented as a weighted sum of embedded constituent words, where weights reflect the positions of words within it.

Sukhbaatar et al. implemented and compared models exploiting those two approaches. For convenience, the error rates of those models are given in Table 1. As could be seen, the model with position encoding showed significantly better results with only 9.4% mean error rate, compared to the 15.4% for the model with BoW representation. Those results are expected, as soon as, BoW representation loses the the ordering of the words in the sentence, but this information might be essential, especially, for the examples based on the subject-object relations.

We also experimented with an approach inspired by work of Sutskever et al.[6] where they introduced an LSTM-based encoder-decoder model for the machine translation problem. This model consists of two multilayered LSTM networks. One works like an encoder, which creates a dense representation of a sentence, given a sequence of constituent words in one natural language. Another works like a decoder, which decodes the sentence from the dense representation to the sequence of the words in another natural language. Sutskever et al. demonstrated that a dense representation of sentences built by LSTM encoder is sensitive to the order of the words but almost insensitive to details like changes from active to passive voice. Such representation might be beneficial for attention mechanism, so we incorporated LSTM encoder into End-to-end Memory Network.

Thus, in our experimental model with LSTM encoding, the transformation of the story fact or question into the internal dense representation has following workflow:

- Convert constituent words of the sentence into the embedded form, using matrices A, B or C (Figure 1).
- Feed the sequence of words in the embedded form into LSTM cell.
- Use final hidden state of LSTM cell as a dense representation of the sentence.

4 Dataset

In this work, we have trained our model on the synthetic bAbI dataset introduced by Weston et al. [8]. This dataset consists of 20 independent tasks, where each task is intended to test one kind of behaviour and looks like a bunch of quadruplets (story, question, answer, supporting facts). Thus, supervision information is provided in the form of correct answers to the questions and set of relevant sentences, there the latter could be ignored in models with soft supervision. According to [8], all tasks are designed to be natural and plain for humans and do not require background knowledge (Figure 2). So, an average human is supposed to achieve

1. John picked up the apple.
2. John went to the office.
3. John went to the kitchen.
4. John dropped the apple.

Q: Where was the apple before the kitchen?
A: Office.

Figure 2: An example of Task 3 "Three Supporting Facts" of bAbI dataset.

100% accuracy on those tasks.

5 Experimental Setup

All models were trained on the bAbI dataset with 10k training examples for each task, where 10% of examples were reserved for cross-validation. To make results comparable with the results of Sukhbaatar et al. [5], the memory size was restricted to the 50 latest sentences of the story. As soon as only 2.4% of the stories in dataset do not fit to this memory size, models had an access to all necessary contextual information for the vast majority of the examples. Subsequent enlargement of the memory size led to the significant increase of training time, but did not influence noticeably the accuracy of the networks.

In one-hot word representation, 0-value was reserved for the nil-word, and all sentences were padded with it to the same size. The embedding matrices were initialized randomly with the dimensionality of embedded space equal to 30xW, where W is vocabulary size (maximum 227 for the full bAbI dataset). All models have been trained with a learning rate $\eta = 0.01$ and annealing of $\eta/2$ every 15 epochs.

Model with LSTM encoding had one LSTM cell with a size of hidden state equal to 96. Dropout of input and output connections (both with keep probability 0.6) was used for regularization.

6 Experimental results

In this work, we compared several approaches to the sentence encoding in order to figure out how the representation of information can influence the performance of a model. The exploited approaches are BoW, position and LSTM encodings, described in Section 3. All models were trained for each task separately. The final results are presented in Table 1

As mentioned, the results achieved by Sukhbaatar et al.[5] for models with BoW and position sentence encodings are provided as well in Table 1 on the left side.

Task	MemN2N [5]		Experimental MemN2N		
	BoW	PE	BoW	PE	LSTM
1: 1 supporting fact	0.0	0.0	0.0	0.0	0.0
2: 2 supporting facts	0.6	0.4	0.5	0.8	35.3
3: 3 supporting facts	17.8	12.6	16.4	14.9	16.6
4: 2 argument relations	31.8	0.0	32.1	0.0	0.0
5: 3 argument relations	14.2	0.8	13.6	0.4	5.3
6: yes/no questions	0.1	0.2	0.3	0.0	5.4
7: counting	10.7	5.7	10.2	2.9	7.3
8: lists/sets	1.4	2.4	4.3	2.4	0.5
9: simple negation	1.8	1.3	1.1	1.8	5.4
10: indefinite knowledge	1.9	1.7	3.5	3.4	15.6
11: basic coreference	0.0	0.0	13.3	12.3	12.6
12: conjunction	0.0	0.0	0.0	0.0	0.0
13: compound coreference	0.0	0.1	6.2	7.4	6.4
14: time reasoning	0.0	0.2	4.2	0.1	7.5
15: basic deduction	12.5	0.0	46.6	0.0	0.0
16: basic induction	50.9	48.6	53.8	55.5	51.9
17: positional reasoning	47.4	40.3	49.7	49.6	32.2
18: size reasoning	41.3	7.4	40.6	7.8	0.8
19: path finding	75.4	66.6	78.4	77.6	79.4
20: agent's motivation	0.0	0.0	0.0	0.0	0.0
Mean Error(%)	15.4	9.4	18.74	11.85	14.11
Failed tasks(err.>5%)	9	6	10	7	13

Table 1: Test error rates for models trained on 10k examples of bAbI dataset. BoW refers to the Bag-Of-Words sentence representation. PE refers to position encoding of sentences. Details of LSTM, BoW and PE encodings are described in Section 3.

For more accurate comparison, we repeated those experiments with our implementation of End-to-end Memory Network (Table 1, right side) and it demonstrated slightly worse mean error rates: 18.74% for BoW and 11.85% for position encodings compared to 15.4% and 9.4% reported by Sukhbaatar et al. In particular, it achieved much worse error rates for tasks 11 ("basic coreference") and 13 ("compound coreference"), but passed the 7th task ("counting"), which was failed by the model of Sukhbaatar et al.

Finally, the comparison of models with three different encoding methods showed, that one with position encoding achieved the best accuracy with mean error rate 11.85% and only 7 failed tasks. The model with LSTM encoding showed smaller

mean error rate (14.11%) than the model with BoW encoding (18.74%). However, the former failed 13 tasks, while the latter failed only 10. Moreover, the training of the model with LSTM encoder required approximately twice more time so in the context of bAbI tasks LSTM encoder seems to be redundant.

7 Conclusion

Due to the discrete nature of memory, the memory-augmented neural networks have to address it as a whole and, thus, require some attention mechanisms to emphasize facts, needed for the generation of an answer. Thus ideally, facts and questions should be represented in the embedded form (analogous to the embedded form of words), which enables computationally inexpensive analysis of semantic proximity.

In this work, our goal was to show how the representation of data in the memory bank can influence the performance of memory-augmented neural networks. For this, we compared three models of End-to-end Memory Network with different dense representations of sentences, namely, Bag-of-Words, position and LSTM encodings. While first two types of encoding are simply the sums of constituent words transformed to the embedded form (though, position encoding is a weighted sum), LSTM encoding is supposed to build a grammar-independent representation of sentences [6].

Results of our experiments showed, however, that position encoding was the most advantageous. Model with that type of sentence representation achieved best results with mean error rate 11.85% against 18.74% and 14.11% demonstrated by others. Moreover, it required significantly less time for training, compared to the model with LSTM encoding, and did not have to be carefully fine-tuned.

Nevertheless, LSTM encoder proved its viability as a practical method of sentence embedding showing comparably good results in our experiments. It does not require a special type of training so that it could be easily incorporated into another model and trained in its context. Thus, we presume it to be a convenient and promising approach to the sentence vectorization. In future work, we are planning to analyze if an output of such encoder could be improved further by adding additional LSTM cells or exploiting another training techniques.

References

[1] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[2] Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[3] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[4] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[5] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.

[6] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[7] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 707–712, 2015.

[8] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[9] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.