CMU OAQA at TREC 2016 LiveQA: An Attentional Neural Encoder-Decoder Approach for Answer Ranking

Di Wang and **Eric Nyberg**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{diwang,ehn}@cs.cmu.edu

Abstract

In this paper, we present CMU's question answering system that was evaluated in the TREC 2016 LiveQA Challenge. Our overall approach this year is similar to the one used in 2015. This system answers real-user submitted questions from Yahoo! Answers website, which involves retrieving relevant web pages, extracting answer candidate texts, ranking and selecting answer candidates. The main improvement this year is the introduction of a novel answer passage ranking method based on attentional encoder-decoder recurrent neural networks (RNN). Our method uses one RNN to encode candidate answer passage into vectors, and then another RNN to decode the input question from the vectors. The perplexity of decoding the question is then used as the ranking score. In the TREC 2016 LiveQA evaluations, human assessors gave our system an average score of 1.1547 on a three-point scale and the average score was .5766 for all the 26 systems evaluated.

1 Introduction

Similar to the inaugural running of the LiveQA track in 2015 [1], the main objective of LiveQA 2016 was to provide automatic answers for real-user questions in real time. The test collection was simply questions that freshly submitted to the Yahoo! Answers website and have not been previously answered by humans. Participant systems need to return answer texts with no more than 1000 characters in length and within 1 minute. System responses were then judged by TREC assessors on a 4-level Likert scale.

CMU's Open Advancement of Question Answering (OAQA) group continued the work from last year's LiveQA submission. We improved the real-time web-based Question Answering (QA) system, and submitted a run to 2016 evaluation. Our QA pipeline begins with candidate passages retrieval and extraction, then answer passages ranking and tiling. In this paper, we focus on discussing our recent development on the answer passages ranking module. Being considered as a key challenge of developing effective QA system, the answer passages ranking and selection is to identify the answer-bearing passages from all candidate passages. The selected passage should contains useful information, and answer the input question. Since there is no official training corpus associated with the challenge, our approach leveraged the vast amount of previously-answered questions from Yahoo! Answers that are available online. During the official run, our QA server received one question per minute for 24 hours and provided answers within one minute for 94% of the input questions. On a normalized three-point average score metric, CMU-OAQA received a score of 1.1547, which was significantly higher than the average score of 0.5766 over all systems. In the rest of this paper, we describe the OAQA LiveQA system in more details.

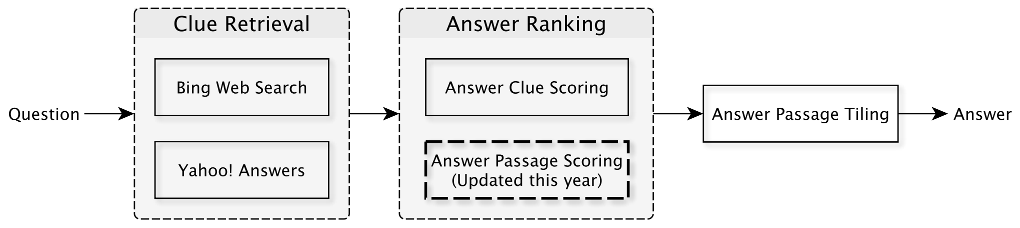

Figure 1: Architecture of the CMU-OAQA LiveQA system

2 Architecture

Our overall system architecture remains the same to the one used in 2015 [2]. The pipeline is briefly described here for completeness, please refer to our last year's report for a more complete description. As illustrated in Figure 1, the architecture of our system decomposes the solution into three major processing phases:

1. **Clue Retrieval**. Given a question title and its full text description, we formulate search engine queries and issue them to different search engines (Bing Web Search, Yahoo! Answers) in order to retrieve web pages related to the question.
2. **Answer Ranking**. Answer candidates (title/body/answer tuples that represent either conceptional questions or answer texts) are extracted from web pages, and ranked based on a relevance estimator. The most effective relevance estimator we found was a heuristically-weighted combination of: a) optimized BM25 similarity scoring over the title and body texts, and b) a novel attentional encoder-decoder recurrent neural networks model that estimates the relevance of a candidate answer text given a question text.
3. **Answer Passage Tiling**. Finally, a simple greedy algorithm is used to select a subset of highest-ranked answer candidates; these are simply concatenated without further processing in order to produce the final answer.

3 Answer Ranking with Attentional Encoder-Decoder Neural Networks

RNN-based encoder-decoder models have been applied to machine translation and quickly achieved state-of-the-art results [3, 4]. Since RNN models do not depend on any external feature or knowledge, we adopted a such neural encoder-decoder model, and trained it to "translate" from an answer passage to the question. We later used the trained model to provide the relevance score between a question and answer based on the likelihood of their "translation".

In last year's submission, we employed a recurrent neural network based approach [5, 6] that uses a multi-layer stacked bidirectional Long-Short Term Memory (BLSTM) network to sequentially read words from question and answer passages, and then output their relevance scores. Because training this BLSTM model requires both positive and negative examples, we followed the same data preparation procedure described by Surdeanu et al. [7] to generate negative labels by retrieving other answer passages from the collection. However, since the generated labels contain false negatives, the model may potentially learn low weights for instances with false negative training labels and decrease overall performance. On the other hand, training attentional neural encoder-decoder model needs only positive pairs, therefore this model will not suffer from above problem anymore.

A basic encoder-decoder neural translation model from [8] suffers from translating a long source sequence efficiently. This is largely due to the fact that the encoder of this basic approach needs to compress the whole source sequence's information into a vector of a fixed dimensionality. Motivated by this, the attention mechanism enables the decoder to revisits the input sequence's hidden states and dynamically collects information needed for each decoding step. Specifically, our new answer passage ranking model is based on a combination of the models of [3] and [4] that we found to be effective. Here we describe the attention-based neural encoder-decoder model we used for ranking in greater detail.

3.1 Model Structure

In general, our neural encoder-decoder model aims at generating a target sequence $Y = (y_1, \ldots, y_{T_y})$ given a source sequence $X = (x_1, \ldots, x_{T_x})$. Each word in both source and target sentences, x_t or y_t, belongs to the source vocabulary V_x, and the target vocabulary V_y respectively.

First, an encoder converts the source sequence X into a set of context vectors $C = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}\}$, whose size varies w.r.t. the length of the source passage. This context representation is generated using a multi-layered recurrent neural network (RNN). The encoder RNN reads the source passage from the first token until the last one, where

$$\mathbf{h}_i = \Psi\left(\mathbf{h}_{i-1}, \mathbf{E}_x\left[x_t\right]\right). \tag{1}$$

Here $\mathbf{E}_x \in \mathbb{R}^{|V_x| \times d}$ is an embedding matrix containing vector representations of words, and Ψ is a recurrent activation unit that we used the Long Short-Term Memory (LSTM) [9] in our submission.

The decoder, which is implemented as an RNN as well, generates one word at a time, based on the context vectors set returned by the encoder. The decoder's hidden state $\bar{\mathbf{h}}_t$ is a fixed-length continuous vector that updated in the same way as Eq. (1). At each time step t in the decoder, a time-dependent attentional context vector $\mathbf{c}_t$ is computed based on the current hidden state of the decoder $\bar{\mathbf{h}}_t$ and the whole context set C.

This starts by computing the content-based score of each context vector as:

$$e_{t,i} = \bar{\mathbf{h}}_t^\top W_a \mathbf{h}_i. \tag{2}$$

This relevance score measures how helpful the i-th context vector of the source sequence is in predicting next word based on decoder's current hidden state $\bar{\mathbf{h}}_t^\top$. These relevance scores are further normalized by the softmax function:

$$\alpha_{t,i} = \text{softmax}(e_{t,i}) = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T_x} \exp(e_{t,j})}, \tag{3}$$

and we call $\alpha_{t,i}$ the attention weight.

The time-dependent context vector $\mathbf{c}_t$ is then the weighted sum of the context vectors with their attention weights from above:

$$\mathbf{c}_t = \sum_{i=1}^{T_x} \alpha_{t,i} \mathbf{h}_i. \tag{4}$$

With the context vector $\mathbf{c}_t$ and the hidden state $\mathbf{h}_t$, we then combine the information from both vectors to produce an attentional hidden state as follow:

$$\mathbf{z}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]). \tag{5}$$

The probability distribution for the next target symbol is computed by

$$p(y_t = k | y_{<t}, X) = \text{softmax}(\mathbf{W}_s \mathbf{z}_t + \mathbf{b}_t). \tag{6}$$

3.2 Parameter Optimization and Network Setup

If we define all the parameters in this model as θ, training this attention-based model can be done by minimize the negative conditional log-likelihood of the training data

$$\hat{\theta} = \arg\min_{\theta} \sum_{n=1}^{N} \sum_{t=1}^{T_y} -\log p(y_t = y_t^{(n)} | y_{<t}^{(n)}, X^{(n)}; \theta),$$

where the log probability inside the inner summation is from Eq. (6). It is important to note that the ground-truth target words $y_{<t}^{(n)}$ is used as input during training. Because the entire model is end-to-end differentiable, so the gradient of the log-likelihood function with respect to all the parameters θ can be computed efficiently by back-propagation.

	NDCG	MAP@			MRR@		
		2+	3+	4+	2+	3+	4+
Lower Bound	0.3924	0.2225	0.1232	0.0519	0.0800	0.0524	0.0274
Upper Bound	1.0000	1.0000	0.7977	0.4668	1.0000	0.7977	0.4668
BM25	0.5636	0.4307	0.2631	0.1205	0.4303	0.2555	0.1174
Encoder-Decoder	**0.6346**	**0.5124**	**0.3390**	**0.1657**	**0.5645**	**0.3672**	**0.1779**

Table 1: Results on re-ranking submitted answers from TREC LiveQA 2015

Our encoder and decoder RNNs contains two-layer stacked LSTMs. Each layer of LSTM has a memory size of 500. The network weights are randomly initialized using a uniform distribution $(-0.08, 0.08)$, and are trained with the ADAM optimizer [10], with an initial learning rate of 0.002. Gradients were clipped so their norm does not exceed 5. Each mini-batch contains 200 answer and question pairs.

We use the Yahoo! Answers Comprehensive Questions and Answers dataset[1] for training, which contains around 4.4 million Yahoo! Answers questions and their best answers. The words of input sentences were first converted to 300-dimensional vector representations learned from RNN based language modeling tool word2vec [11]. Each passage's beginning and end are also padded with a special boundary symbol, `<S>`.

3.3 Ranking

At test time, instead of finding the best-scoring "translation", the decoder is fed with original question as input, and calculate the perplexity that the model predicts regarding question words:

$$score(X, Y) = \exp(-\frac{1}{T_y}\sum_{t=1}^{T_y} \log p(y_t = y_t^{(n)} | y_{<t}^{(n)}, X^{(n)}; \hat{\theta})).$$

4 Development Set Analysis

The LiveQA organizers provided scored answers from all TREC 2015 LiveQA submissions [2], which we used as a development dataset for analyzing the performance of answer passage ranker. Since we are only evaluating ranker's performance, questions with only negative candidate answer passages are removed from evaluation. This development dataset contains 949 questions. On average, there are 19.5 answer passage candidates for each question and 32% of candidates' score are above "Fair".

In order to validate our new ranker's performance with this dataset, we re-rank above development dataset with our ranker and a BM25 baseline ranker. NDCG (Normalized Discounted Cumulative Gain) with graded relevance scale 0-3, MAP (Mean average precisionk), and MRR (Mean Reciprocal Rank) were then used as evaluation metrics (calculated using the official $trec\_eval$ evaluation scripts).

Table 1 summarizes our preliminary experimental results on the Yahoo! Answers question retrieval and ranking task. Although good performance on this dataset does not necessarily correlate to good performance on the LiveQA 2016 challenge, it does demonstrate the necessity of developing non-trivial candidate retrieval and answer ranking methods for any LiveQA-related task.

5 Official Evaluation Results

In this year's LiveQA evaluation, 1015 questions (out of 1088 submitted questions) were judged and scored using a 4-level Likert scale:

- **4**: Excellent: "a significant amount of useful information, fully answers the question"

[1] `http://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
[2] https://sites.google.com/site/trecliveqa2016/liveqa-qrels-2015/LiveQA2015-qrels-ver2.txt.gz

System ID	Avg score (0-3)	#Answers	Success@ 2+	3+	4+	Precision@ 2+	3+	4+
Avg of all runs	0.5766	771.0385	0.3042	0.1898	0.0856	0.3919	0.2429	0.1080
CMU-OAQA	**1.1547**	**954**	**0.5606**	**0.3951**	**0.1990**	**0.5964**	**0.4203**	**0.2117**

Table 2: Official TREC 2016 LiveQA track evaluation results.

- **3**: Good: "partially answers the question"
- **2**: Fair: "marginally useful information"
- **1**: Bad: "contains no useful information for the question"
- **-2**: "the answer is unreadable (only 15 answers from all runs in 2015)"

The evaluation measures used are:

- **avg-score (0-3)**: "average score over all queries (transferring 1-4 level scores to 0-3, hence comparing 1-level score with no-answer score, also considering -2-level score as 0)"
- **succ@i+**: "number of questions with i+ score (i=1..4) divided by number of all questions"
- **prec@i+**: "number of questions with i+ score (i=2..4) divided by number of answered only questions"

Table 2 summarizes the results of our system run and average scores from all submitted runs. We believe the overall performance of our system to be encouraging, as it suggests that our system can provide a useful answer (fair, good, or excellent) for more than 56% of the questions.

6 Conclusion and Future Work

This paper presented our improvements and evaluation results for our LiveQA 2016 system. Although this system performed significantly better than average of the 26 systems evaluated, the low absolute evaluation values indicate that there is still much room for improvement. In the future, we want to further utilize redundancy in answer passage for better answer ranking and tiling.

References

[1] Eugene Agichtein, David Carmel, Dan Pelleg, Yuval Pinter, and Donna Harman. Overview of the TREC 2015 liveqa track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*, 2015.

[2] Di Wang and Eric Nyberg. CMU OAQA at TREC 2015 liveqa: Discovering the right answer with clues. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC 2015, Gaithersburg, Maryland, USA, November 17-20, 2015*, 2015.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[4] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421, 2015.

[5] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *Annual Meeting of the Association for Computational Linguistics*, pages 707–712, 2015.

[6] Di Wang and Eric Nyberg. A recurrent neural network based answer ranking model for web question answering. In *SIGIR Workshop on Web Question Answering: Beyond Factoids*, 2015.

[7] Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

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

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.