CMU OAQA at TREC 2015 LiveQA: Discovering the Right Answer with Clues

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 automatic, web-based, real-time question answering (QA) system that was evaluated in the TREC 2015 LiveQA Challenge. This system answers real-user questions freshly submitted to the Yahoo! Answers website that have not been previously answered by humans. Given the title and body of the question, we generated multiple sets of keyword queries and retrieved a collection of web pages based on those queries. Then we extracted answer candidates from web pages in the form of answer passages and their associated clue. Finally, we combined both IR- and NLP-based relevance models to rank and select answer candidates. In the TREC 2015 LiveQA evaluations, human assessors gave our system an average score of 1.081 on a three-point scale, the highest average score achieved by a system in the competition (the second-best score was .677, and the average score was .465 for the 21 systems evaluated).

1 Introduction

LiveQA is an emerging TREC challenge for automatic, real-time, user-oriented question answering (QA) systems. Real user questions are selected as inputs from a stream of newly submitted questions on the Yahoo! Answers site [1], which have not yet received a human answer. As per the requirements for this track, participants must deploy their systems as web services which provide answers to questions in real time (within one minute). An additional requirement is that the answer text should be no more than 1000 characters in length. System responses are judged by TREC assessors on a 4-level Likert scale.

Attempting to answer user-generated questions in real time is an exciting but difficult challenge. The questions generated by real users are often ambiguous, ungrammatical and vary greatly in length, topic(s) and language style(s). Unlike previous TREC QA tracks, there is no pre-defined question type (factoid, list, or definition) associated with the question/answer pairs in the LiveQA challenge. There is also no constraint on what the user may ask about, which implies that there is no pre-defined answer type; much like the Jeopardy! challenge, we must deal with "lexical answer types" [1].

CMU's Open Advancement of Question Answering (OAQA) group developed a real-time web-based QA system for the LiveQA challenge in 2015. Since there is no official training corpus associated with the challenge, our approach leveraged the vast amount of text data that is available online, especially previously-answered questions from Yahoo! Answers. We also designed and implemented a new data model and novel relevance ranking methods for LiveQA. During the official run, our QA web service received one question per minute for 24 hours and provided answers within one minute for 97.9% of the input questions. On a normalized three-point average score metric, the

[1] https://answers.yahoo.com/

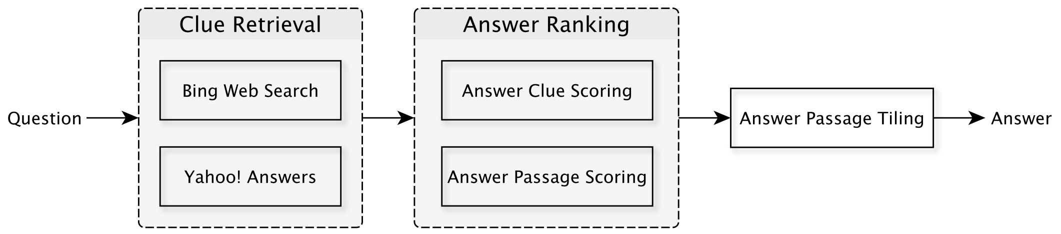

Figure 1: Architecture of the CMU-OAQA LiveQA system

CMUOAQA received a score of 1.081, which was the top score in the 2015 LiveQA evaluation (the second-best average score was .677, and the average over all system runs was .467). In this paper, we describe the OAQA LiveQA system in more detail.

2 Approach

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 recurrent neural network approach 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.

In the remainder of this section, we describe each phase and sub-component in greater detail.

2.1 Input

Each question q transmitted to a LiveQA web service consists of four parts: title q_title, body q_body, category, and a unique question ID. Our system does not make use of the question category, and only uses the question ID to filter out retrieved Yahoo! Answers pages with the same ID.

2.2 Clue Retrieval

Given a question, firstly we formulate multiple queries, then send them to the search engines, thirdly collect best matching web pages for each, and finally harvest a set of candidate answers for further processing.

The input question $q = \langle q\_title, q\_body \rangle$ is converted to sets of keyword queries. In order to achieve higher recall, we generate multiple sets of keywords from q_title, q_body, and the concatenation of both. However, the inclusion of full q_body text may result in very long queries, for which the search engines do not return any result. Therefore we also generate keyword queries with only informative noun phrases, bigrams, and unigrams. The different keyword queries are then sent to search engines for web retrieval. We collect the snippets returned for each hit and also download the full text of the web page for each hit.

Our system utilizes Bing's web search API[2] to retrieve answer candidates from the Web. Since Shtok et al. [2] showed that a significant amount of the unresolved Yahoo! Answers questions can

[2]`https://datamarket.azure.com/dataset/bing/searchweb`

be satisfactorily answered by reusing a best answer from the past, our system also uses the search function on Yahoo! Answers to collect additional answer candidates.

Our goal here is to search the Internet and select passages of text, containing information relevant to the question and indicating the presence of a candidate answer passage on the same page. For this purpose, we build a general data structure called the *answer clue* (a_clue), represented as $\langle c\_title, c\_body \rangle$. This formulation can be used to represent (and effectively combine) either the title and body of a similar question on community QA (cQA) sites like Yahoo! Answers, or a document title and search snippet extracted from a web page. Finally, we heuristically select answer passage text $a\_passage$ based on its relative position to a_clue. Our algorithm favors passages using heuristics like: a) the voted best answer on cQA sites, b) the first reply on Internet forums, and c) paragraphs appearing in contexts which highly match the search snippet.

2.3 Answer Ranking

Each candidate answer $a = \langle a\_clue, a\_passage \rangle$ is weighted by estimating how well both the a_clue and $a\_passage$ matches the question. The relevance ranking score between input question q and an answer candidate a is then defined as:

$$S(q, a) = S_c(q, a\_clue) \times (1 + w_p \times S_p^{'}(q, a\_passage))$$

where S_c and $S_p^{'}$ are answer clue score and normalized answer passage scores respectively, and w_p is the weight factor to merge them. The original answer passage score S_p is normalized by the max and min values (S_p^{max} and S_p^{min}) in the collection of answer passage scores generated for the current input question:

$$S_p^{'} = \frac{S_p - S_p^{min}}{S_p^{max} - S_p^{min}}$$

Question q and answer clue a_clue both contain two fields: title and body. Intuitively the title field is more concise and informative than the body field. The answer clue score S_c is, then, computed as weighted sum of retrieval scores of all text and title only text:

$$S_c(q, a\_clue) = S_{bm25}(q\_title \oplus q\_body, c\_title \oplus c\_body) + w_t \times S_{bm25}(q\_title, c\_title)$$

where w_t (set to 1.0) is the weight to boost the title field matching score, and S_{bm25} is the Okapi BM25 [3] formula (with parameters $K_1 = 1.0$ and $B = 0.75$). The idf and average document length values that are required to compute BM25 were obtained by indexing the Yahoo! Answers Comprehensive Questions and Answers dataset[3], which contains around 4.4 million Yahoo! Answers questions and their answers.

To integrate the term proximity information in our relevance estimation, we also use the sequential dependence variant of the Markov random field model [4] to formulate weighted queries to the BM25 function. The weights for unigram, bigram, and proximity are 0.8, 0.1, and 0.1 respectively. We also expanded unigram queries with synonyms from WordNet [5] when the query word had only one sense in WordNet.

To calculate the relevance score S_p between question and answer passage, we employ a recurrent neural network based approach [6, 7] that uses a multilayer stacked bidirectional Long-Short Term Memory (BLSTM) network to sequentially read words from question and answer passages, and then output their relevance scores. Figure 2 illustrates how we use the stacked BLSTM to model the answer passage ranking and selection problem. The words of input sentences were first converted to vector representations learned from the RNN-based language modeling tool word2vec [8]. In order to differentiate q_title and $a\_passage$ sentences, we inserted a special symbol, `<S>`, after the question sequence. Then, the question and answer sentence word vectors are sequentially read by BLSTM from both directions. In this way, the contextual information across words in both question and answer sentences is modeled by employing temporal recurrence in BLSTM.

We used the same experiment settings described by Wang and Nyberg [6] to train this model, except that the training dataset was switched to a random subset of 100k questions from the Yahoo! Answers Comprehensive QA dataset. In order to adapt this dataset for model training, we follow the same data

[3]`http://webscope.sandbox.yahoo.com/catalog.php?datatype=l`

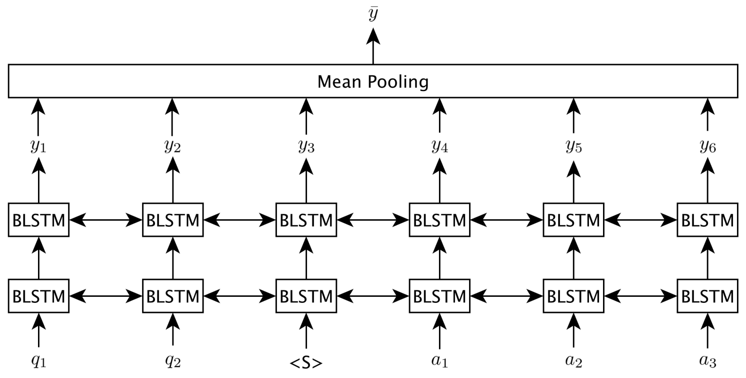

Figure 2: Answer passage ranking model based on stacked BLSTM

preparation procedure described by Surdeanu et al. [9] 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. To avoid this we set the combination weight $w_p = 0.1$ for this year's evaluation, and plan to re-train the model and tune w_p in the future using the questions and labeled responses from this year's LiveQA evaluation.

2.4 Answer Passage Tiling

Given a ranked list of answer passages, the system must produce an optimal text that answers the question. Good answers vary in length from short to long, and may contain a short reference to a specific entity or factoid, or a long narrative explanation; in general, the ideal answer length varies with the user's information need (as expressed by the question). This year, we simply assumed that longer answers are preferred.

If the top-scoring answer passage was longer than 500 characters, then the first 1000 characters (LiveQA length constraint) are directly returned as the final answer text. If not, we applied an answer tiling algorithm, which assembles longer answer texts out of shorter answer passages. The algorithm proceeds greedily from the top-scoring answer passages to all subsequent candidates whose ranking score is higher than 90% of the highest score (top decile). Additional answer passages are appended to the final answer text, with a prefix to indicate the start of each new answer passage; e.g. "Opinion 2:" for the second passage, and so on. The tiling algorithm terminates when the length of the generated answer passage is greater than 50% of the maximum allowable length (1000 characters).

3 Development Set Analysis

The LiveQA organizers provided a sample of 1000 Yahoo! Answers question IDs (QIDs), which we used as a development set for pre-evaluation analysis. Specifically, we used each question's title text as an input query, and then collected outputs from the Yahoo! Answers default "Search Answer" function. In particular, a list of returned QIDs was collected via "Relevance" search (instead of "Newest", "Most Answers", or "Fewest Answers" search). The first 100 search results for each query were collected [4] and automatically labeled [5] by comparing their source question IDs with input question IDs.

In order to validate our system's performance with this dataset, we processed the same query set with our retrieval and ranking modules, and returned a ranked list of question IDs. MRR (Mean Reciprocal Rank) and P@1 (Precision at one) were then used as evaluation metrics (calculated using the official *trec_eval* evaluation scripts).

[4]This dataset is available at `https://github.com/yuvalpinter/LiveQAServerDemo/tree/master/data/1k-ya-search-results`

[5]Note that the generated labels will also contain false negatives.

	Set Recall	P@1	MRR
Yahoo! Answers default search	0.8464	0.5873	0.6434
Clue Retrieval + Answer Ranking	**0.9100**	**0.8950**	**0.9008**

Table 1: Results on development dataset

Run ID	Avg score	Success@				Precision@		
	(0-3)	1+	2+	3+	4+	2+	3+	4+
CMU-OAQA	**1.081**	**0.979**	**0.532**	**0.359**	**0.190**	**0.543**	**0.367**	**0.195**
Avg of all runs	0.465	0.925	0.262	0.146	0.060	0.284	0.159	0.065

Table 2: Official TREC 2015 LiveQA track evaluation results.

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 2015 challenge, it does demonstrate the necessity of developing non-trivial candidate retrieval and answer ranking methods for any LiveQA-related task.

4 Official Evaluation Results

In this year's LiveQA evaluation, 1,087 questions (out of 1,340 submitted questions) were judged and scored using a 4-level Likert scale:

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

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 53% of questions.

5 Conclusion and Future Work

This paper presented the overall system architecture and individual phases and components for our LiveQA 2015 system[6]. Although this system performed significantly better than average for the 21 systems evaluated, the low absolute evaluation values indicate that there is still much room for improvement. One promising future direction is to utilize the scored answer passages from this year's evaluation to train supervised models for keyword generation, answer passage extraction and answer passage ranking. We can also extend the current pipeline by incorporating an answer confidence estimation to detect and prune low-quality final answers. Another potentially interesting challenge is to develop scalable evaluation methods that approximate the TREC assessor judgments

[6]`https://github.com/oaqa/LiveQA`

in a more efficient and cost-effective way (through the use of crowd-sourcing), making it possible to develop much larger labeled datasets for supervised learning. We also intend to focus on more sophisticated models for answer tiling which a) combine answer passage relevance, uniqueness, and conciseness to select final answers, and b) consider post-processing of selected answer passages to make final answers more concise and readable.

Acknowledgments

This research was supported in part by the InMind Intelligent Agents project funded through a generous gift from Yahoo!. We would also like to thank Dan Pelleg, David Carmel, and Eugene Agichten for insightful discussions.

References

[1] Adam Lally, John M. Prager, Michael C. McCord, Branimir Boguraev, Siddharth Patwardhan, James Fan, Paul Fodor, and Jennifer Chu-Carroll. Question analysis: How watson reads a clue. *IBM Journal of Research and Development*, 56(3):2, 2012.

[2] Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. Learning from the past: Answering new questions with past answers. In *Proceedings of the 21st International Conference on World Wide Web*, pages 759–768, 2012.

[3] Stephen E. Robertson and Steve Walker. On relevance weights with little relevance information. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 16–24, 1997.

[4] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 472–479, 2005.

[5] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38 (11):39–41, 1995.

[6] 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.

[7] 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.

[8] 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.

[9] 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.