Real, Live, and Concise: Answering Open-Domain Questions with Word Embedding and Summarization

Rana Malhas[1], Marwan Torki[1], Rahma Ali[1], Evi Yulianti[2], Tamer Elsayed[1]
[1] Department of Computer Science and Engineering, Qatar University, Qatar
{rana.malhas, mtorki, rahma.ali, telsayed}@qu.edu.qa
[2] School of Computer Science and IT, RMIT University, Australia
{evi.yulianti}@rmit.edu.au

ABSTRACT

Resorting to community question answering **(CQA)** websites for finding answers has gained momentum in the recent years with the explosive rate at which social media has been proliferating. With many questions left unanswered on those websites, automatic question answering (QA) systems have seen light. A main objective of those systems is to harness the plethora of existing answered questions; hence transforming the problem to finding good answers to newly-posed questions from similar previously-answered ones or composing a new concise one from those potential answers. In this paper, we describe the real-time Question Answering system we have developed to participate in TREC 2016 LiveQA track. Our QA system is composed of three phases: answer retrieval from three different Web sources (Yahoo! Answers, Google Search, and Bing Search), answer ranking using learning to rank models, and summarization of top ranked answers. Official track results of our three submitted runs show that our runs significantly outperformed the average scores of all participated runs across the entire spectrum of official evaluation measures deployed by the track organizers this year.

Keywords

LiveQA; real-time question answering; learning to rank

1. INTRODUCTION

The ubiquitous presence of community question answering (CQA) websites has motivated research in the direction of building automatic question answering (QA) systems that can benefit from previously-answered questions to answer newly-posed ones [8]. A core functionality of such systems is their ability to retrieve and effectively rank previously-suggested answers with respect to their degree/probability of relevance to a posted question. The ranking functionality is vital to push away irrelevant and low quality answers, which is commonplace in CQA as they are generally open with no restrictions on who can post or answer questions.

TREC 2016 LiveQA track is in succession for the second year in a row. It focused on "live" question answering for real-user questions. Real user questions, sampled from the stream of most recent questions submitted on the Yahoo! Answers site that have not been answered by humans, are sent live to the participating systems. The questions are sampled from a list of seven categories that include Arts & Humanities, Beauty & Style, Health, Home & Graden, Pets, Sports and Travel. Each system should provide an answer in "real time", not exceeding 1 minute in response; and the length of each answer should not exceed 1000 characters.

The LiveQA track organizers have provided a training dataset of questions and answers previously judged on a 4-point scale, ranging from poor to excellent, by the judges of TREC 2015 LiveQA track. The questions in this dataset are those sent to the QA systems of participating teams in TREC 2015 LiveQA track; and the answers are those returned by the participating teams. Henceforward, we shall refer to this dataset as the QRELS dataset.

In this paper, we describe the question answering system we have developed to participate in TREC 2016 LiveQA Track. Our QA system is composed of three phases: answer retrieval, answer ranking, and summarization.

1. The answer retrieval phase retrieves answers from three different sources: an archive of Yahoo! Answers (YA) recent questions and their answers, Bing search, and Google search.

2. The answer ranking phase adopts a supervised learning approach. We train a learning-to-rank (L2R) model over labeled QRELs data acquired from the previous TREC 2015 LiveQA track. We use the learned model to rank candidate answers retrieved from all sources. The features we use is a mix of traditional and nontraditional features such as language model/query likelihood and word embedding features generated from the GloVe (Global Vectors for Word Representation)[1] pretrained model, respectively. A thorough description of each feature is presented in section 2.2.1.

3. Finally, we apply an optional summarization module on the top three ranked answers to presumably formulate a best and concise answer. For summarization, we followed [2] by using the model proposed by Takamura

TREC'16 Gaithersburg, Maryland USA

[1] http://nlp.stanford.edu/projects/glove/

and Okumura [11] to generate extractive summaries from the top-ranked passages.

Given an incoming question, the retrieval module retrieves top candidate answers from one or all the above mentioned three sources. The ranking module in turn ranks those candidate answers; and then the summarization module is optionally applied to summarize the top ranked answers to return a best answer, otherwise the system returns the top ranked answer as the best answer. We have used MAP (Mean Average Precision) as the evaluation measure in our pre-submission experiments to evaluate the effectiveness of our QA system.

The rest of the paper is organized as follows: the approach and system design are introduced in section 2; the experimental evaluation and setup followed by our submissions to the LiveQA Track; finally, we conclude with final remarks.

2. APPROACH AND SYSTEM DESIGN

Our question answering system was designed with high modularity; it is composed of three modules: an answers retrieval module, an answers ranking module and an optional summarization module; they are described in sections 2.1, 2.2, and 2.3, respectively. The training pipeline of the system is illustrated in Figure 1 and an overview of the system is depicted in Figure 2.

The retrieval module has leveraged previously-indexed Yahoo! Answer questions and answers, as well as Web resources such as Bing and Google. The ranking module tackled the answer ranking task with a supervised learning approach that leveraged learning-to-rank models. The features used in training those models were a mix of traditional and nontraditional features as described in section 2.2.1. Details regarding the retrieval methods, specific models and features used in our three submissions to TREC 2016 LiveQA Track are presented in section 3.

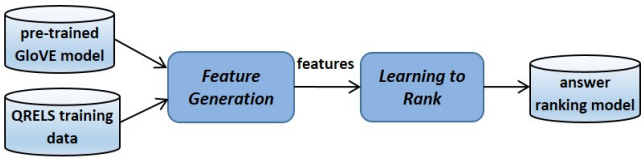

Figure 1: Training Pipeline

2.1 Answer Retrieval

The answer retrieval module is composed of three components (a component for each source); the system can work with any combination of these components; this makes our system highly extensible should we need to incorporate more answer sources. Although the components are currently pipelined, we are considering to parallelize them in future versions of our system to optimize the response time.

2.1.1 Yahoo! Answers Search

The Lucene indexer is responsible for indexing questions and their answers from two datasets in offline mode. The first dataset (12 GB) was originally crawled from Yahoo! Answers during July 2005 till December 2006 and released online[2] by Yahoo! Research for research purposes; it is composed of about 4.4 million questions and all their answers.

[2] http://webscope.sandbox.yahoo.com/

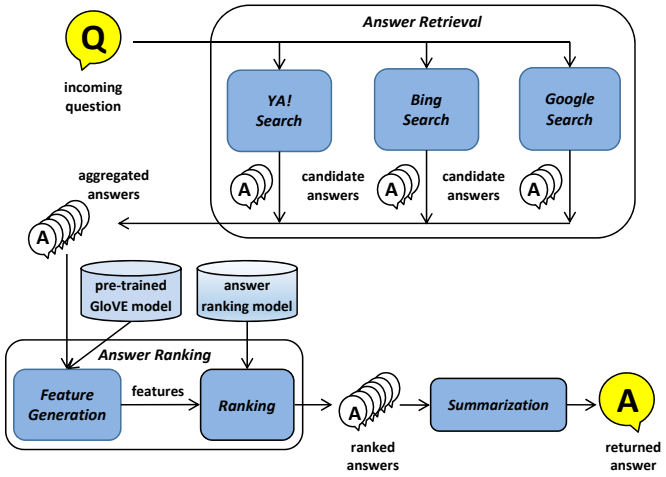

Figure 2: System Overview

The second dataset (50 GB) was also crawled from Yahoo! Answers; it covers the years 2013 through 2016. Only questions that had a best answer were indexed along with all their other answers. The resulting index contains 16.9 million previously answered questions with a total size of 23.3 GB.

Given an incoming question, the Lucene index searcher (with its default retrieval model) is responsible for retrieving the best answers of the top 10 matching question titles; i.e., question-question similarity was adopted.

2.1.2 Bing Search

This component is responsible for submitting an incoming question as a query using the Bing search API methods and retrieving the top 5 search results. Each search result constitutes a title, link, text and primary snippet. Each search result link is accessed and the HTML of the webpage returned is parsed to locate the primary snippet. Three cases are considered depending on the location of the primary snippet. If it is found in the body tag of the HTML webpage, the 1000 characters appearing right after the primary snippet are saved as a secondary snippet and returned as a candidate answer. Otherwise, if the primary snippet is from the meta-data description of the webpage, the first 1000 characters from the body tag of the HTML webpage are saved as a secondary snippet and returned as a candidate answer. Finally, if the primary snippet is not in the webpage, the "<Snippet not found in HTML>" string is saved as the secondary snippet and the primary snippet is returned as a candidate answer instead.

2.1.3 Google Search

This component is responsible for submitting an incoming question as a query to Google.com. It then parses the results page returned to retrieve the top 5 search results. Each search result constitutes a title, link, text and primary snippet. Each search result link is accessed and the HTML of the webpage returned is parsed to locate the primary snippet. The same three cases described for the Bing component above are adopted for returning candidate answers.

2.2 Answer Ranking

The answer ranking module is composed of two components; a feature generation component, and a ranking component that comprises the ranking models and answers scoring.

2.2.1 Feature Generation

We have opted to utilize covariance word embedding features and four of the Metzler-Kanungo (MK) features [6]; the MK features are computed at the answer level and not at the sentence level as computed in the original paper.

- **Covariance Word Embedding** To compute this feature, we define a document d as the concatenation of an incoming question and a candidate answer. Every document $d_n \in D$, where $n \in \{1, \cdots, N\}$, has a set of words. Each word has a fixed-length word embedding representation, $w \in \mathbb{R}^{Dim}$, where Dim is the dimensionality of the word embedding. Thus for every document d_n in the set D, we define $d_n = \{w_1, \cdots, w_{k_n}\}$, where k_n is the number of words in the document d_n. The word embedding representation is computed offline following Mikolov et al approach [7].

  For a document that has k_n words, we compute the covariance matrix $C \in \mathbb{R}^{Dim \times Dim}$. The covariance matrix C is computed by treating each dimension as a random variable and every entry in $C_{n_{u,v}}$ is the covariance between the pair of variables (u, v). The covariance between two random variables u and v is computed as in eq. 1, where k_n is the number of observations (words).

$$C_{n_{u,v}} = \frac{\sum_{i=1}^{k_n}(u_i - \bar{u})(v_i - \bar{v})}{k_n - 1} \tag{1}$$

  The matrix $C_n \in \mathbb{R}^{Dim \times Dim}$ is a symmetric matrix. We compute a vectorized representation of the matrix C_n as the stacking of the lower triangular part of matrix C_n as in eq. 2. This process produces a vector $v_n \in \mathbb{R}^{Dim \times (Dim+1)/2}$

$$v_n = \left\{C_{n_{u,v}} : u \in \{1, \cdots, Dim\}, v \in \{u, \cdots, Dim\}\right\} \tag{2}$$

- **Metzler-Kanungo Features** We have implemented four out of the six features used by Metzler and Kanungo [6], and applied them at the answer level instead of the sentence level:
  - **AnswerLength:** Number of terms in an answer, after stop-words are removed.
  - **ExactMatch:** Equals 1 if there is an exact lexical match of the question string occurring in the answer, and 0 otherwise.
  - **TermOverlap:** Fraction of question terms that occur in the answer after stop word removal and stemming.
  - **LanguageModelScore:** Language model score is computed as the log likelihood of the question terms being generated from the answer. The answer language model is smoothed using Dirichlet smoothing [12]. It can also be referred to as query likelihood language model score [4]. This score is computed as in eq. 3.

$$f_{LM}(Q, A) = \sum_{w \in Q} tf_{w,Q} \log \frac{tf_{w,A} + \mu P(w|C)}{|A| + \mu} \tag{3}$$

    where $tf_{w,Q}$ is the number of times the word w occurs in the query (question), $tf_{w,A}$ is the number of times that w occurs in the answer, $|A|$ is the length of the answers, $P(w|C)$ is the background language model and μ is a parameter for Dirichlet smoothing. We tend to set μ as 1000; i.e towards the lower end of the range recommended by TREC which is 1000-2000 because answers are shorter in length than documents.

2.2.2 Ranking Models

A learning-to-rank (L2Rank) setup was adopted that is similar to [3, 5, 9]. The L2Rank models were trained over the labeled QRELS dataset provided by the LiveQA track organizers. The data constituted 1,087 questions and their potentially related answers (18 answers per question on average) making a total of 20,469 labeled answers. Each answer is labeled by either being *poor*, *fair*, *good* or *excellent* with respect to its question. The labels were the outcome of the judging process on a 4-point scale by the judges of TREC 2015 LiveQA Track. The questions in this dataset are those sent to the QA systems of participating teams in TREC 2015 LiveQA Track; and the answers are those returned by the 2015 participating teams.

The main algorithm used for training our learning-to-rank models that were deployed in our three submissions to the LiveQA track was the MART (Multiple Additive Regression Trees, a.k.a. Gradient boosted regression tree) algorithm.

2.2.3 Answers Scoring

Upon receiving the computed features for an incoming question and its retrieved candidate answers, the trained L2Rank model is utilized to score these answers before returning them sorted in descending order using their attained scores to our QA server (system). If summarization is to be applied, the top three scored answers are forwarded to the summarization module; otherwise, only the top scored answer is returned by the QA server as the best answer.

2.3 Summarization

Upon receiving the top three scored answers for the given question, we used the model proposed by [11] to generate an extractive summary. The model solves the maximum coverage problem where the coverage of important terms is maximized while the redundancy is minimized. The model is transformed into a linear programming problem. It has been shown in [2] that adding more statements to the summary was better than returning only one statement.

3. EXPERIMENTAL EVALUATION

In this section we present the experimental setup of our three submitted runs to the TREC 2016 LiveQA Track, and report their official results. All three submissions achieved above average scores across the spectrum of official performance evaluation measures described in section 3.3.

3.1 Experimental Setup

Table 1: Pre-submission experiments comparing MAP scores of models trained using the MK feature set, and Covariance word Embedding (CovWE) features over a 5-fold cross validation data setup; suffix numbers 50 and 100 designate the dimensionality of the vectors representing the word embedding. Best MAP score is boldfaced.

Feature Set	Features	MAP with 5-Fold CV
MK	ExactMatch AnswerLength TermOverlap LanguageModel	**0.4705**
MK & Word Embedding	MK + CovWE-50	0.3180
	MK + CovWE-100	0.3260

We have used the labeled QRELS dataset of 1087 questions and their potentially related 20,469 answers provided by the LiveQA track organizers in training our models. For our three submissions, the MART (Multiple Additive Regression Trees, a.k.a. Gradient boosted regression tree) algorithm was the main algorithm used in training the L2Rank models. The models differed in the features used in their training. We evaluated those models using 5-fold cross validation over the MAP (Mean Average Precision) metric as the objective function. Among the most prominent preprocessing steps conducted on the QRELS dataset was mapping the 4-point scale labels ranging from 1-4 to the range 0-3. This mapping was necessary to get a more reliable evaluation of the L2Rank models. RankLib[3] was used to create and evaluate our learning-to-rank models.

The pre-trained GloVe[4] (Global Vectors for Word Representation) word vectors of 50 and 100 dimensions were used in generating the covariance word embedding feature as described in section 2.2.1.

3.2 Pre-Submission Experiments

In our preliminary experiments to train and evaluate the L2Rank models, we have adopted an incremental strategy in the features used. We started with the 4 Metzler-Kanungo (MK) features in isolation before adding the word embedding features; the covariance word embedding (CovWE) features of 50 dimensions were added first, making a total of 1279 features; then they were replaced by the CovWE features of 100 dimensions, thus making a total of 5054 features. The MART algorithm was used in training all the models.

Surprisingly, the model trained over the MK feature set outperformed the other two models trained over all features, including the covariance word embedding of 50 dimensions in one model, and 100 dimensions in the other model (Table 1). This finding prompted us to conduct post-submission experiments that provided enough evidence to affirm that feature concatenation might not be the best way to combine the CovWE and MK features; and different approaches might be more effective in combining them.

[3] https://sourceforge.net/p/lemur/wiki/RankLib/

[4] http://nlp.stanford.edu/projects/glove/

An observation worth noting is that the covariance word embedding features CovWE-50 and CovWE-100 with the MK feature set have attained comparable MAP scores of 0.3182 and 0.3260 in Table 1, respectively. As such, we decided to use the simpler model of CovWE-50 in one of our submissions (QU2) instead of using CovWE-100.

3.3 Performance Evaluation Measures

For the TREC 2016 LiveQA Track, each question with the corresponding pool of answers was examined and evaluated by TREC editors. The answers were judged and scored using a 4-level scale (with 1 signifying 'Bad and non-useful' to 4 signifying an 'Excellent answer') based on their relevance and responsiveness to the question. The main metrics include: average answer score, precision (fraction of answered questions with a score above a threshold), and coverage (fraction of all questions answered)[5]. The MAP (Mean Average Precision) metric was the main evaluation measure we used in our pre-submission experiments.

3.4 Submissions to the TREC 2016 LiveQA Track

Every participating team was allowed three submissions to the LiveQA track. Each of our submissions was a variant version of our QA system/server as explained below.

Our preliminary feature extraction experiments revealed that the L2Rank models trained over the MK feature set outperformed all the other models trained so far (Table 1). For this reason this feature set was used in all submissions, either in isolation or with word embedding features. Also the MART algorithm was used in training all the models. The main differences among submissions lie in the set of features used in training the L2Rank models, the sources used for answer retrieval, the size of the Lucene index used, and whether summarization was applied or not. We shall refer to the three submissions as QU, QU2, and QU3, respectively.

QU submission. All components of the answer retrieval module were utilized; namely the Yahoo! Answer index, Bing search and Google search. But only the index of the 12 GB dataset crawled from Yahoo! Answers by Yahoo! Research was used. The size of this index amounted to 8.2 GB comprising 4.4 million questions and their answers. As for the answer ranking module, the 4 MK features and MART algorithm were used in training the L2Rank model. Summarization was not applied.

QU2 submission. Again, all components of the answer retrieval module were utilized; namely the Yahoo! Answer index, Bing search and Google search. Here, the full sized Lucene index of 23.3 GB that comprised 16.9 million questions was used. As for the answer ranking module, all features including the MK and covariance word embedding (of 50 dimensions) features along with the MART algorithm were used in training the L2Rank model. Summarization was applied. For some cases when the summarizer failed to return an answer, the first ranked answer was returned using the trained learning to rank model.

QU3 submission. The answer retrieval module only utilized the full sized Yahoo! Answer index (23.3 GB compris-

[5] https://yahooresearch.tumblr.com/post/138611916336/call-for-participation-trec-2016-liveqa

Table 2: Official results of our run submissions to LiveQA track compared to the average of all submitted runs. Best scores are boldfaced

Evaluation Measure	QU Run	QU2 Run	QU3 Run	Track Average
avg score (0-3)	0.7842	0.8768	**0.9005**	0.5766
succ@2+	0.4236	**0.4670**	0.4631	0.3042
succ@3+	0.2532	0.2956	**0.2975**	0.1898
succ@4+	0.1074	0.1143	**0.1399**	0.0856
prec@2+	0.4419	**0.5011**	0.4667	0.3919
prec@3+	0.2641	**0.3171**	0.2999	0.2429
prec@4+	0.1120	0.1226	**0.1410**	0.1080
Questions Answered	973	946	**1007**	771.0385

ing 16.9M questions). But the web search components (Bing search and Google search) were not used. As for the answer ranking module, the MK feature set and MART algorithm were used in training the L2Rank model. Summarization was not applied.

It is worth noting that the MK feature values in the QU and QU3 submissions were linearly normalized; i.e each feature was normalized by its min/max values.

Table 2 shows that the best performing run among our submissions, across all evaluation measures, was the QU3 submission with an average score of 0.9005, followed by QU2 (0.8768) and QU (0.7842) submissions.

Figure 3 exhibits examples of best answers returned by the system (from three different sources) to three LiveQA track questions.

3.4.1 Discussion

Contemplating over the results of our submissions led us to the following observations.

- Our best performing submission (QU3) is quite similar to the TREC 2015 LiveQA submission by Bagdouri and Oard [1] in that they both used a Yahoo! Answer index as the sole source in answer retrieval. However, our QU3 submission achieved an average score of 0.900 as compared to 0.615 by Bagdouri and Oard [1] despite the fact that we only used the crawl of the last four years and not the full crawl of Bagdouri and Oard [1]. The difference in performance could be attributed to two reasons; our system used learning to rank over (MK) features while [1] did not. Moreover, we indexed questions and their answers only if a best answer was prevalent to the question.

- Our QU3 submission this year marked a significant improvement over our TREC 2015 LiveQA submission last year [10] which was also based on a Yahoo! Answer index as well. The main differences lie in the index size used (only 250K documents were used last year) and the deployment of an answer ranking module this year.

- Our expectations towards the performance of the QU submission was high because the answer retrieval module leveraged the three sources (Yahoo! Answer index, Bing and Google search), but with a smaller index size to increase the chances of getting best answers from the web. However, the performance of the QU submission was the least among our submissions which could be attributed to the time-out incidences encountered while searching the web.

Question: Foot pain unable to walk?

Best answer: X-rays and other imaging tests (CT scans, MRIs) may be ordered if there are not obvious signs of an underlying cause, or if you are unable to put weight on your foot. Physical examinations may also reveal where the problem is located in the foot. Treatment Options for Your Foot Pain There are numerous things you can do to control or treat your foot pain. The first step is seeing a foot specialist, a podiatrist, or a foot and ankle surgeon. You can sometimes help yourself by using the acronym RICE to relieve foot pain. RICE stands for Rest, Ice, Compression, and Elevation.

Source: Bing...http://www.footvitals.com/problems/foot-pain.html. ***Run:*** QU

Question: Do you need an exotic animal permit to own a peacock?

Best answer: You need a licence to keep some animals, eg: wild cats primates wild dogs, eg wolves certain pigs, eg wild boar marsupials See the full list of animals you need a licence for. Hybrid or cross-bred animals may need a licence, depending on how far removed the animal is from its wild ancestor. If you're not sure, ask your local council. Last updated: 12 November 2014 …

Source: Summarized from mixed sources. ***Run:*** QU2

Question: Which is an example of the absolutism under peter the great?

Best answer: fighting the great northern war required unprecedented economic resources, and russia's yawning budget deficit and aging infrastructure meant that the state could not effectively allocate resources and money in wartime. peter's government was constantly in dire need of money, and at first it responded by monopolizing certain strategic industries, such as salt, alcohol, oak, and tar. peter also taxed many russian cultural customs (such as bathing, fishing, beekeeping, and wearing beards) and issued tax stamps for paper goods. however, with each new tax came new loopholes and new ways to avoid them, and so it became clear that tax reform was simply not enough. **the solution was a sweeping new poll tax, which replaced a household tax on cultivated land. previously, peasants had skirted the tax by combining several households into one estate; now, however, each peasant was assessed individually for a tax of 70 kopeks, paid in cash.

Source: Yahoo! Answer Index. ***Run:*** QU3

Figure 3: Examples of best answers returned by the QA system to three LiveQA questions from variant sources

- Likewise, our expectations towards the performance of the QU2 submission was even higher than QU and QU3 because it used the MK and word embedding features in addition to leveraging the three sources for answer retrieval. Moreover, the summarization module was applied as well. We suspect that the time-out issues encountered during web search to be the main reason for attaining a relatively lower average score of 0.877 as compared to QU3 (0.900).

- The number of answered questions in QU2 submission is less than the number of answered questions in QU3. We projected the score of QU2 on the number of answered questions by QU3. We found that score of QU2 would be 0.9333 if we answered 1007 questions like in QU3.

4. CONCLUSION

This paper describes the question answering system we developed to participate in TREC 2016 LiveQA Track. Our system was developed with high modularity to make it easily extensible. It is composed of three modules: an answers retrieval module, an answers ranking module and an optional summarization module. The answer retrieval module constitutes three components to retrieve candidate answers from a Yahoo! Answer index, Bing search and Google search. For the answer ranking module we have adopted a

supervised learning approach where learning-to-rank models were trained over four of the Metzler-Kanungo (MK) features and covariance word embedding features generated from the pre-trained GloVe (Global Vectors for Word Representation) model. The labeled QRELS dataset provided by the LiveQA track organizers was used in training the L2Rank models. We extracted summaries from top ranked answers using our learning to rank model.

Three submissions were made to the LiveQA track; each constituting a variant version of our question answering system. The main differences among submissions lie in the set of features used in training the L2Rank models, the sources used for answer retrieval, the size of the Yahoo! Answer index used, and whether summarization was applied or not. Our three submissions achieved above average scores of all submitted runs to the LiveQA track, and across all the official evaluation measures used by TREC. The best performing run among our submissions was the one using the Yahoo! Answer index of 16.9M YA questions for answer retrieval (without Bing and Google), a L2Rank model trained over MK features for answer ranking, and without applying summarization. The next to best run (among our submissions) used all three sources (Yahoo! Answer index, Bing and Google) for answer retrieval, a L2Rank model trained over covariance word embeddings (of 50 dimensions) and MK features for answer ranking, in addition to applying summarization on the top three ranked answers. The least performing run among our submissions also used the three sources for answer retrieval, but it used the L2Rank model trained over MK features only; and without summarization.

More experiments are needed on our QA system to evaluate variant configurations of the three modules. Our participation in the LiveQA track was an eye opener to the evolving and promising potential of automatic and smart question answering systems that can harness the plethora of previously answered questions.

5. ACKNOWLEDGMENTS

This work was made possible by NPRP grant# NPRP 6-1377-1-257 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

We thank Mossaab Bagdouri for providing the crawled database of Yahoo! Answers questions and their corresponding available answers.

6. REFERENCES

[1] M. Bagdouri and D. W. Oard. CLIP at TREC 2015: Microblog and LiveQA. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC*, pages 17–20, 2015.

[2] R.-C. Chen, J. S. Culpepper, T. T. Damessie, T. Jones, A. Mourad, K. Ong, F. Scholer, and E. Yulianti. RMIT at the trec 2015 liveqa track. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC*, 2015.

[3] R.-C. Chen, D. Spina, W. B. Croft, M. Sanderson, and F. Scholer. Harnessing semantics for answer sentence retrieval. In *Proceedings of the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 21–27. ACM, 2015.

[4] W. B. Croft, D. Metzler, and T. Strohman. *Search engines: Information retrieval in practice*, volume 283. Addison-Wesley Reading, 2010.

[5] R. Malhas, M. Torki, and T. Elsayed. QU-IR at SemEval 2016 Task 3: Learning to rank on Arabic community question answering forums with word embedding. *Proceedings of SemEval*, pages 866–871, 2016.

[6] D. Metzler and T. Kanungo. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*, pages 40–47, 2008.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[8] A. Shtok, G. Dror, Y. Maarek, and I. 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. ACM, 2012.

[9] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers on large online qa collections. In *ACL*, volume 8, pages 719–727, 2008.

[10] R. Suwaileh, M. Hasanain, M. Torki, and T. Elsayed. QU at TREC-2015: Building real-time systems for tweet filtering and question answering. In *Proceedings of The Twenty-Fourth Text REtrieval Conference, TREC*, 2015.

[11] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics, 2009.

[12] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.