Automated Question Answering System

Chanin Pithyaachariyakul, Maria Khvalchik, Anagha Kulkarni
Department of Computer Science
San Francisco State University
1600 Holloway Ave, San Francisco
CA, USA, 94132
cpithyaa@mail.sfsu.edu, mkhvalch@mail.sfsu.edu, ak@sfsu.edu

Abstract

We present *SFS-QA*, an automatic real-time question-answering system, that can answer free-text questions within a minute. *SFS-QA* system analyzes a question and transforms it into a boolean keyword query using Stanford Dependency Parser and MetaMap tagger. The query is used to obtain matching web pages from the World Wide Web through Google Search API. In addition, two specific data sources: Yahoo! Answers and Wikipedia are used for matching the documents. The resulting web pages are mined for candidate answers. Finally, Learning to Rank based LambdaMart, and Recurrent Neural Network based BLSTM algorithms are used to learn, rank, and select the best answer from candidate answers. For empirical evaluation, TREC LiveQA 2015 and 2016 datasets which consist of about 1000 questions each were used. The results demonstrate that our system substantially outperforms strong baselines.

1 Introduction

Even with modern search engines, there are many scenarios where the users struggle to find the information they are looking for. This is especially true when the information need is complex, and when the user is unable to distill down their information need into a few keywords. These factors motivate many users to seek answers on community based Question-Answering (QA) sites, such as, Yahoo! Answers[1], and Quora[2], where the question can be posted in natural language, as opposed to keyword query, and the answer(s) to the question is provided by the members of the online community. Compared to the experience that users have with commercial search engines, the QA sites provide two key advantages: (i) freedom to specify their information need (question) in free flowing natural language, and (ii) convenience of receiving focused answer(s) to the question, as opposed to receiving 10 web pages that have to be read and parsed to identify the answer to the question. On the other hand, one clear advantage of search engines over QA sites is the information response time. The search results are available to the user instantaneously, whereas human-authored answers may take much longer to be posted.

These observations motivate our QA system that allows the user to specify the question in natural language, which in turn is internally transformed into a boolean query composed of key terms and phrases. Our approach works on the premise that the answers for the majority of questions are already available on some web page(s) on WWW. Thus the goal of the next phase of our system is to obtain these web pages, and extract concise answers from the content of the pages. An additional objective that we set for the QA system is to have low latency: every question has to be answered in less than one minute. This requirement influences many design decisions made for the system's architecture.

The key contributions of this work are designing and developing: (i) a light-weight but effective question to query transformation approach (ii) a multi-sourced document retrieval approach, and (iii) a highly effective answer ranking approach. A thorough empirical evaluation of individual phases of the system, and of the end-to-end system was undertaken, which demonstrates that the proposed QA system performs substantially better than strong baselines while meeting the response time requirement.

[1] https://answers.yahoo.com
[2] https://www.quora.com

2 Related Work

Liu et al. analyzed the difference between queries and questions in the context of community QA sites (Qiaoling Liu, 2016). They confirmed the common belief that queries are focused on key concepts (things/nouns and actions/verb). While questions also include contextual and etiquette related terms. In their analysis only 31.4% of terms overlapped between question and query. They also observed that questions contain more abbreviations, and shortened versions of terms, since the writer perceives more freedom of writing with questions.

Savenkov et al. introduced a new QA system that uses multiple data resources such as Wikipedia data, WWW, and Yahoo! Answer Collection (Savenkov and Eugene, 2016). Wikipedia corpus is used to obtain relevant documents for an answer, and results from Web search are used for query expansion. They detected keywords from top snippet results. Then, they expanded keyword terms that may be misspelled and have multiple thesauruses from the snippets. This lexical match increases the opportunities to match more relevant documents when a less popular keywords is used in a query. Finally, they used WebScope L6 dataset, which contains 4.4 million Yahoo! Answer questions, to run through a supervised learning to label the association between question terms and answer entities. The label is used to evaluate the candidate answer based on its association score. Their result showed a significant improvement because web search results are the effective resources that enhance the query understanding. Moreover, the question-answer pairing successfully ranked candidate answers by helping with entity identification.

Shtok et al. proposed a new approach to answer a new question from CQA site such as Yahoo! Answers by reusing past answers from previous similar questions from CQA site itself (Shtok and Szpektor, 2012). They applied a cosine similarity to match potential past questions as the candidate selections. The next step is to extract only the best answer from the selected old questions as candidate answers. Then they applied statistic classifier to select the final answer. This research achieved high precision answering and preserved human generated content answer unlike other automatic question answering systems that used web extraction to generate the answers.

Pinter et al. introduced a new method that applied a grammatical dependency parser to identify segments of CQA questions to generate queries (Pinter and Reichart, 2016). Because CQA questions are often long and verbose, the dependency parser is required to partition a question into several syntactically independent segments. The segment queries, which are generated from fragments of a question, are more effective to find the relevant answers than the simple phrase queries.

Soricut and Brill (R. and E., 2006) published one of the first papers on non-factoid question answering, and many others have followed (R. and H.; Surdeanu et al., 2011; Oh J. H., 2012). As a training set they used a corpus of 1M question-answer pairs from FAQ collected on the Web. To search for the answer candidates they used MSNSearch and Google. Our work uses different algorithm for QFM, is trained using Yahoo! Answers dataset and uses learning to rank techniques which started to advance in mid-00s. In recent years the advancements in NLP/ML techniques and availability of large QA datasets have propelled research and contests on answering open-domain non-factoid questions (Agichtein E., 2015). Wang et al. (Wang and Nyberg, 2015, 2016) works were the winner of two subsequent TREC LiveQA competitions. In the first paper they trained an answer prediction model using BLSTM Neural Network. In the second - Neural Machine Translation techniques to train the model which generates the answer itself given only a question. We use their method as a baseline comparing our work against to.

3 SFS-QA: An automated real-time question-answering system

Our QA system is structured as a pipeline of four components: Query formulation, Document retrieval, Candidate answer extraction, and Answer selection. The first module is responsible for parsing, and transforming the question into a query, that is used by the second phase to retrieve documents that may contain the answer, the third phase is tasked with identifying the minimal unit of text that answers the original question, and the fourth phase selects the unit of text that serves best as the answer to the question. The architecture of the system is shown in Figure 1.

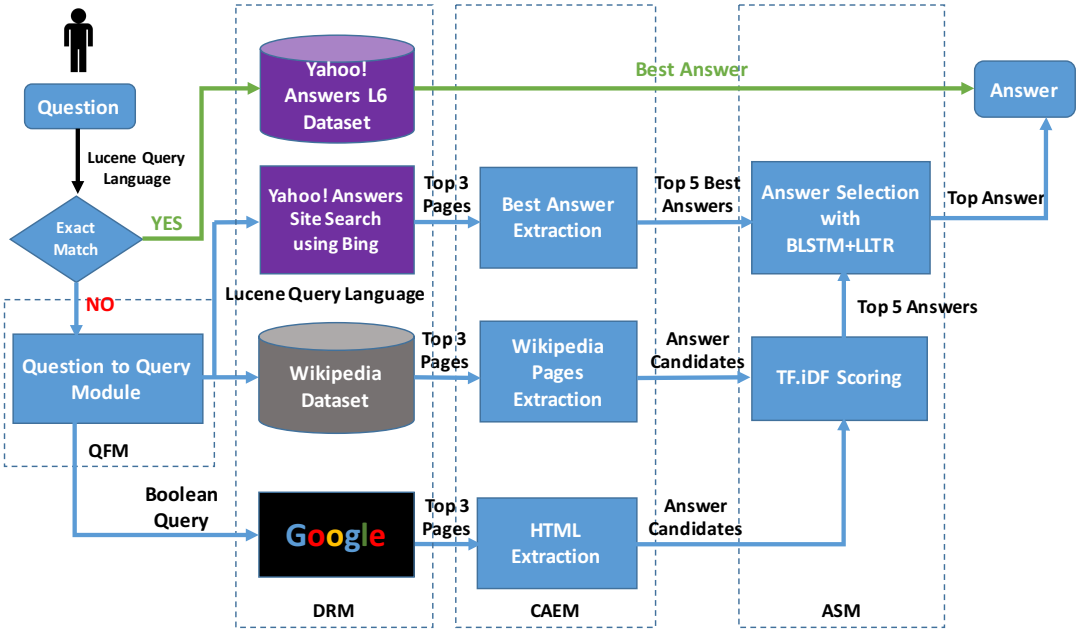

Figure 1: System Architecture for SFS-QA system. QFM: Query Formulation Module. DR: Document Retrieval Module. CAEM: Candidate Answer Extraction Module. ASM: Answer Selection Module.

3.1 Query Formulation Module (QFM)

This module transforms free-text questions to a well-formed boolean conjunctive queries that can be evaluated by a search engine. This is a challenging problem because questions are often verbose. Questions often contain information that is useful for a human reader but is superfluous, or even misleading, if included in the search query. We address this verbosity problem at multiple levels.

First, not every sentence in the question contributes to the final query. Only sentences that start with WH-words (e.g. Who, When, Where, Why) and end with a question mark do (Varanasi and Neumann, 2015). Second, within a sentence only certain select parts of the question are included in the query. Also, the length of these parts, individual words or phrases, is selected carefully. For example, transforming the following question, *"Why's juice from orange peel supposed to be good for eyes?"*, into a unigram boolean query: *(orange) AND (peel) AND (juice) AND (good) AND (eyes)*, is not effective because most of the retrieved web-pages are about *orange juice* and not about *orange peel juice*. In order to construct a boolean query that retains the key information in the question, QFM performs detailed grammatical analysis of the question. Specifically, we use the Stanford Dependency Parser (Chen and Manning, 2014) to identify the various phrases (noun, verb, preposition, and adjective phrases) in the sentence. This allows us to identify important phrases, rather than just individual words. For the above question, this approach selects important phrases and generates the final boolean conjunctive query as follows: *(juice) AND (orange peel) AND (good for eyes)*. This query is successful at retrieving web-pages about *orange peel juice* rather than about *orange juice* even though the latter has the more dominant presence on the web.

The English *closed class* terms (pronouns, determiners, prepositions) in the question are often ignored since they do not capture the central topic of the question. However, in certain situations the prepositions should be included in the query. In case of the following question, *"How much should*

I pay for a round trip direct flight from NYC to Chicago in early November?", if the preposition words, *from* and *to*, are ignored then the information about the travel direction is lost. Preposition detection is used to address this issue where the grammatical tree structure of the sentence is leveraged to identify the preposition phrases, such as, *from NYC* and *to Chicago*, and these are included as-is in the boolean query.

The *verb phrase* is another important dependency phrase that the system needs to identify because sometimes single verb term is too broad, and thus not useful in retrieving relevant documents. However, in such circumstances the verb phrase provides a focused search query. As an example question, *"How to map dowse using a pendulum?"*, without the verb phrase detection, the system generates the query: *(map) AND (dowse) AND (use) AND (pendulum)*. Once the query is run through a search engine, it might retrieve a distorted set of documents because the verb *map* is ambiguous between mapping either *dowse* or *pendulum*. The verb phrase detection, however, generates a more explicit query: *(map dowse) AND (use pendulum)* that is less likely to retrieve ambiguous result set.

All of the above transformations are necessary when the question is verbose. However, when the question is well-articulated and succinct, no transformations are performed. Questions with 8 words or less are considered concise, and used as search queries as-is.

A substantial fraction of the posted questions on the community sites are related to *health*. These questions also tend to have certain unique properties, such as, the larger *vocabulary gap* between the question and the content of the relevant documents. A vocabulary gap exists between two units of text when they use different vocabulary to convey the same meaning. Since the questions are authored by ordinary people they tend to use the common names for diseases, conditions, and symptoms, while the relevant documents written by medical professionals/experts tend to use the technical names for these concepts. As a result, a document that is relevant to the question might have very little word overlap with the question, and thus not be retrieved. To avoid this, query expansion is often proposed as the solution. Doing effective query expansion for medical text is a non-trivial problem. Fortunately there exists an excellent resource, MetaMap tagger[3], that we use in our work for health category questions (Aronson and Lang, 2010). For example, the question, "*how to treat type 2 diabetes without medication"*, is transformed into the following query using the synonyms suggested by the MetaMap tagger for this question, *(medication OR pharmaceutical preparations) AND (non-insulin-dependent) AND (type 2 diabetes OR diabetes mellitus)*.

In summary, the input to the Query Formulation Module is the user question, and QFM transforms it into a query, which is handed over to the next phase.

3.2 Document Retrieval Module (DRM)

The goal of this module is to use the generated query to obtain a set of web pages that are likely to contain answer(s) for the question. First, however, it is checked if the current question has already been answered on Yahoo! Answers. If that is the case then the answer that has been voted as the best answer is returned by the system. Finding an exact match of the question on Yahoo! Answers is however a rare occurrence.

A targeted search is conducted on two online knowledge sources: English Wikipedia, and Yahoo! Answers. To facilitate faster query response time, we maintain a local copy of the English Wikipedia, and the query is run against this local copy. The top three wiki pages returned for the query are added to the set of answer-bearing web pages. We use Solr/Lucene to index and search the Wikipedia copy.

For Yahoo! Answers (Y!A), we use a two-pronged strategy to provide short query response time. A local copy is maintained of the Webscope L6 dataset [4], which is a snapshot of the Y!A site captured in October 2007. This data consist of 4.4+ million questions and all the posted answers, along with metadata, such as, question category, and best voted answer. This data is also indexed with Solr/Lucene for efficient access and search. In addition to L6, the system is also capable of conducting a site search of Yahoo! Answers, in order to obtain the most up-to-date data. The *site search* functionality of Bing Search API is used to accomplish this. It was found that Google Search API is biased against Yahoo! Answers, and thus Bing was chosen for this task. The top three an-

[3] https://metamap.nlm.nih.gov

[4] https://webscope.sandbox.yahoo.com

swer pages are added to the set of answer-bearing web pages.

Finally, the larger World Wide Web is searched using the Google Search API. The top three web pages returned for the query are added to the set of answer-bearing web pages. In total, this module identifies at most nine web pages that are passed on to the next module.

3.3 Candidate Answer Extraction Module (CAEM)

The set of web pages identified by the Document Retrieval Module are downloaded, and each page is passed through the following text processing pipeline. The first step extracts ASCII text from the web page using an html2text library[5]. We refer to the extracted text as a document. This document is next split into *passages*, where each passage consists of four consecutive sentences, the most popular answer length in Yahoo! Answers dataset. A sliding span of four consecutive sentences is used to generate the passages. Thus, a document containing five sentences would generate two passages. This approach generates many passages, specifically, $1 + (n - 4)$, where n is the total number of sentences in the document. The passage length of four sentences was chosen based on data. On an average, high quality answers in the L6 dataset contain four sentences. Shorter answer lengths (single sentence) are common for factoid questions but majority of the L6 questions cannot be categorized as purely factoid.

Passages that do not contain any of the query terms, or that contain more than 2 line breaks, or more than 10 punctuation marks, or non-printable symbols are eliminated. Also, passages that are not in English are filtered out. The langdetect library[6] is employed for language identification. All the passages that survive the filtering step are considered as *candidate answers*.

3.4 Answer Selection Module (ASM)

In this final step of the QA pipeline, the best answer from all the candidate answers is chosen. We experiment with three algorithms for this task: (i) Learning To Rank (LeToR) based LambdaMart algorithm (Burges, 2010), (ii) Neural Network based BLSTM algorithm (Graves and Schmidhuber, 2005), and (iii) a combination approach that employs both, LambdaMart and BLSTM.

There is a rich history of LeToR approaches being applied to automated QA (Bilotti et al., 2010; Surdeanu et al., 2011; Agarwal et al., 2012). Following on this tradition, for the baseline approach, we employ the LambdaMart algorithm to learn a ranking model for scoring the candidate answers, and the highest scored answer is selected as the final answer. We refer to this answer selection approach as LLTR. A subset of the *Webscope Yahoo! Answers L6* dataset[7] is used for training the LLTR model. For many questions in this dataset one of the answers for the question is identified as the best answer. For training LLTR the best answer is assigned the highest rank label, and the remaining answers are assigned a rank label proportional to their BM25 score with the best answer. The following feature set is computed for each <question, answer>pair: Okapi BM25 score, cosine similarity, number of overlapping terms, number of punctuation marks in the passage, number of words in the answer, number of characters in the answer, query likelihood probability, largest distance between two query terms in the answer, average distance between two terms, number of terms in longest continuous span, maximum number of terms matched in a single sentence, maximum number of terms in order. Before computing each of these features, all terms from query and candidate answer were stemmed using Porter.

Recurrent Neural Network (RNN) based approaches have received a lot of attention from the QA community recently (Severyn and Moschitti, 2015; Cohen and Croft, 2016; Wang and Nyberg, 2015, 2016). Since carefully feature engineering is completely unnecessary for NNs these networks lend themselves very well to the QA problem where it is difficult to defining features that generalize well. In fact, the best performing system (Encoder-Decoder) at the TREC 2016 LiveQA track employed a recurrent neural network based approach. In our work we have employed the Bidirectional Long Short Term Memory (BLSTM) neural network because it adapts well to data with varying dependency spans length. The bidirectional property of this network allows for tracking of both, forward and backward relations in the text. We use a modification of network architecture implemented in (Wang and Nyberg, 2015). The network consists of several layers: the word embedding layer followed by BLSTM layer,

[5] https://pypi.python.org/pypi/html2text
[6] https://pypi.python.org/pypi/langdetect
[7] http://webscope.sandbox.yahoo.com

dropout layer to reduce overfitting, mean pooling, and dense layer for the output. The output for the network is a number from 0 to 1 identifying how likely the answer matches the question. It was trained with *ADAM* optimizer, with binary cross-entropy as a target loss function. To train the network a subset containing 384K <question,answer >pairs from the *Webscope Yahoo! Answers L6* dataset was used.

The third answer selection approach that we investigate simply combines the above two approaches. The score assigned by BLSTM to each <question, answer>pair is used as an additional feature in the feature set used by the LLTR ranking algorithm.

4 Results and Analysis

We conduct a thorough empirical evaluation of the individual components of our system, and of the end-to-end system. The results of these evaluations are presented in this section.

4.1 Query Formulation Module

Evaluation Data: Recall that QFM module is tasked with *understanding* the question, and compiling a set of web pages (URLs) that are likely to contain the answer(s) to the question. Evaluation dataset for this task is not readily available. We had to thus re-purpose the annotated datasets that are available for another task – question-answer evaluation. Specifically, we used the TREC LiveQA 2015 and 2016 datasets, which consist of 1000 questions each. For each question there are one or more answers, and each answer is assessed for its relevance to the question by a human, and assigned a score between 0 (non-relevant) and 3 (very relevant). The LiveQA 2016 dataset provides the source URL for each answer. For the LiveQA 2015 dataset we had to locate the source URL for each answer, since it is not included in the dataset. This gives us an annotated set of <Question,URL,score>tuples which we use to evaluate the effectiveness of QFM.

Baselines: The LiveQA 2015 and 2016 questions consist of three fields, title, body, and category. Since these questions are generated by real users the question can be either in title or in description. Based on this observations we have defined three baselines to compare with the proposed QFM:

1. Original-Q: This baseline tests the minimalist approach where no processing is performed on the question. The question is used as query as-is. Here the intuition is that since the questions are authored by humans, no information should be filtered out or added into the original question. As such, the question title, along with the body field is used as the query.

2. QuestionMark-Q: This baseline works with the assumption that humans often add superfluous details to the question, and these details typical occur in sentences that do not end with question mark. As such, sentences in title and body fields, that end with question mark are used as the query.

3. Unigram-Q: This last baseline seeks to filter out unnecessary information further by removing stopwords, and by applying morphological normalization using Krovetz stemmer to sentences ending with question mark in title and body fields. The terms that remain are treated as unigrams and compiled into a boolean AND query.

Evaluation Metrics: For the task at hand, the generated query is said to be effective if it can retrieve one or more answer-bearing web pages in the first three ranks, since these pages are mined for candidate answers in the next phase. As such, this is a precision oriented task, and thus inspires our choice of evaluation metrics: Precision@ranks1,2,3 and NDCG@ranks1,2,3. The latter models the different grades of relevance that are present in the annotation scores.

Results & Analysis: Table 1 reports the results for the three baselines and QFM with and without site search of Yahoo! Answers. It should be noted that URLs for which human scores are not available are considered non-relevant in this assessment. This is an important point that leads to overall low values seen in Table 1.

The prominent trend in these results is that QFM outperforms all the baselines substantially across all the metrics. The results for the Original-Q baseline demonstrate that using the user question as-is often does not lead to relevant web pages. This justifies the need of a Query Formulation Module. The QuestionMark-Q results show that filtering out the sentences without question mark improves the performance by reducing

Table 1: Results of QFM

	Precision@			NDCG@		
	1	2	3	1	2	3
TREC 2015						
Original-Q	0.030	0.027	0.020	0.048	0.056	0.048
QuestionMark-Q	0.046	0.037	0.027	0.078	0.083	0.069
Unigram-Q	0.032	0.028	0.024	0.062	0.064	0.062
QFM	0.073	0.064	0.055	0.159	0.143	0.125
TREC 2016						
Original-Q	0.043	0.035	0.029	0.044	0.057	0.065
QuestionMark-Q	0.064	0.046	0.041	0.069	0.072	0.090
Unigram-Q	0.055	0.043	0.035	0.058	0.064	0.078
QFM	0.106	0.083	0.063	0.179	0.165	0.158

Table 2: Results of ASM

	NDCG	MAP@			MRR@		
		2	3	4	2	3	4
TREC 2015							
Encoder-Decoder	0.635	0.512	0.339	0.166	0.565	0.367	0.178
LLTR	0.622	0.484	0.316	0.155	0.549	0.352	0.156
BLSTM	0.656	0.546	0.347	0.174	0.587	0.379	0.205
LLTR + BLSTM	0.660	0.550	0.349	0.176	0.590	0.381	0.206
TREC 2016							
LLTR	0.648	0.512	0.346	0.217	0.621	0.381	0.241
BLSTM	0.671	0.559	0.379	0.254	0.648	0.403	0.288
LLTR + BLSTM	0.675	0.567	0.384	0.257	0.650	0.399	0.293

the unnecessary information and noise. But filtering out stopwords, and loosing the grammatical structure of the question completely is detrimental, which is demonstrated by the Unigram-Q results. As is described in Section 3.1, QFM attempts to balance two conflicting objectives: (i) distill down the question to the bear minimal query (ii) keep the necessary units of the question intact. These results indicate that this intuition behind QFM is correct.

4.2 Answer Selection Module

Evaluation data: To evaluate the performance of ASM, we use the questions TREC 2015 LiveQA evaluation set. Since answers are rated by judges, it is possible to check how good our system is at ordering of these answers.

Baselines: For the baseline, we use the results obtained by Encoder-Decoder system which was the best performing system of 2016 year. They were testing their version of ASM on 2015 data, and therefore we can compare their system to ours on this dataset (Wang and Nyberg, 2016). The performance of their ASM module on 2016 data is not available, we only have access to the results of the overall system performance which we compare to ours later.

Evaluation Metrics: The effectiveness of the system was evaluated using standard evaluation: NDCG (Normalized Discounted Cumulative Gain), MAP (Mean average precision) at rank X, and MRR (Mean Reciprocal Rank) at rank X.

Results & Analysis: Table 2 provides the results for the evaluation of the ASM. LLTR is less effective than the state-of-the-art approach across all the metrics. However, the neural network based approach, BLSTM performs substantially better than Encoder-Decoder and LLTR for both datasets. The results for LLTR+BLSTM illustrate that two approaches have complementary strengths that can be combined to obtain the best results for the task. The difference between LLTR and LLTR+BLSTM is statistically significant.

The results for MAP show that in 55% of the

cases our system selects the answer with at least fair quality (2+), and in 21% of the cases the quality is excellent. The performance on TREC 2016 is better than on TREC 2015. For reference, Table 2 provides the performance of the TREC 2016 LiveQA winning system – Encoder-Decoder (Wang and Nyberg, 2016), but our system outperforms it. Interestingly, a simple LLTR which relies only on statistical text features is very close in performance to Encoder-Decoder neural network approach. It implies that LLTR is a powerful method with right selection of the features.

We believe that quality of the model can be improved by sanitizing the training dataset. Currently, two main problems are: (i) presence of words with misspellings which make computations of statistical features imprecise; (ii) quality of the best answers manually selected by voters. There exist a few approaches to diminish impact of both issues such as (Chen et al., 2007) for misspellings and (Agichtein et al., 2008) for keeping only high-quality answers.

4.3 End-to-end System Performance

Evaluation data: For the end-to-end system evaluation, the existing datasets cannot be reused because the system generated answers can be different from the previously annotated answers. Thus, we undertook the task of manual assessment of the answers generated by our system. Each answer was rated on the same scale as in TREC LiveQA competition which is 0 (non-relevant) through 3 (very relevant). We selected at random 100 questions from LiveQA 2015, and 100 from LiveQA 2016, and assessed their respective answers on the 4-point scale.

Baselines: We compare our results with those produced by the winning systems at TREC LiveQA 2015, and 2016, the Open Advancement of Question Answering (OAQA) (Wang and Nyberg, 2015), and the Encoder-Decoder (Wang and Nyberg, 2016) system, respectively. Both systems are similar - they use Yahoo! Answers and Bing Web Search for candidate answers retrieval. The only difference between them is the strategy of best answer selection. The former system uses a type of BLSTM while the latter uses an encoder-decoder recurrent neural network model.

Evaluation Metrics: We use the official metrics used by TREC LiveQA track. The *succ@X+* is a fraction of answers with grade above X, where X=annotation_score + 1. The *avgScore* is the average score of the answers produced by a system, where answer scores' range is 0-3.

Results & Analysis: The results are presented in Table 3. Our system performed substantially better than the state-of-the-art for both 2015 and 2016. This is especially prominent at higher score levels, indicating that our system is able to generate better quality results. For 2015, the system produced 53% of answers which could be considered as fair, 40% questions as good, and 25% questions were considered as excellent answers. For the 2016, 67% of answers have a fair grade, 45% are good and 23% are excellent. The average scores, although higher than those of state-of-the-art, indicate that there is plenty of room for improvement.

On an average, the system generates an answer for a submitted question in less than half a minute on a non-specialized commodity computer. The Query Formulation Module transforms the question into a query in 3 seconds, the Document Retrieval Module and the Candidate Answer Extraction Module together take 8 seconds to generate all the candidate answers, and the last module of Answer Selection needs 15 seconds to select the best answer, which amounts to 26 seconds per query, on average.

Table 4 presents examples of the user questions, the corresponding queries, and the answers generated by SFS-QA system. The first example includes an instance of query expansion using MetaMap. The scientific name of *bats*, *Chiroptera*, is sourced from MetaMap. The query generated for the second question shows an example of verb phrase and adjective phrase. Other queries also show examples of noun phrases. In general, these examples illustrate how the QFM module retains the important bigrams and phrases while reducing the superfluous terms. The generated answers, more often than not, provide the necessary information, at least partially.

5 Conclusions

We presented an automated question answering system that accepts questions in natural language and responds with a focused answer in less than half a minute. With thorough empirical evaluation we demonstrated that a light-weight question-to-query transformation module can be developed that is also highly effective. We also illustrated

Table 3: Overall Results

	avgScore(0-3)	succ@2+	succ@3+	succ@4+
TREC 2015				
SFS-QA	1.180	0.530	0.400	0.250
OAQA	1.081	0.532	0.359	0.190
TREC 2016				
SFS-QA	1.350	0.670	0.450	0.230
Encoder-Decoder	1.155	0.561	0.395	0.199

that various existing information sources can be leveraged to obtain answer-bearing web pages. Finally, we established that advances in deep learning can be utilized to select the best answer for the question.

References

Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D Lawrence, David C Gondek, and James Fan. 2012. Learning to rank for robust question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pages 833–842.

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *Proceedings of the 2008 international conference on web search and data mining*. ACM, pages 183–194.

et al. Agichtein E. 2015. Overview of the trec 2015 liveqa track. In *Proceedings of TREC.*.

Alan R Aronson and Franois-Michel Lang. 2010. An overview of metamap: historical perspective and recent advances. *Journal of the American Medical Informatics Association: JAMIA* 17(3):229–236.

Matthew W Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank learning for factoid question answering with linguistic and semantic constraints. In *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM, pages 459–468.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11(23-581):81.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP, 2014*.

Qing Chen, Mu Li, and Ming Zhou. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*. Citeseer, volume 7, pages 181–189.

Daniel Cohen and W Bruce Croft. 2016. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*. ACM, pages 143–146.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

et al. Oh J. H. 2012. Why question answering using sentiment analysis and word classes.

Yuval Pinter and Roi Reichart. 2016. Syntactic parsing of web queries with question intent. In *Proceedings of NAACL-HLT 2016, pages 670-680*.

et al Qiaoling Liu. 2016. When web search fails, searchers become askers: Understanding the transition. In *Proceedings of SIGIR 2012*.

Higashinaka R. and Isozaki H. ????

Soricut R. and Brill E. 2006. Automatic question answering using the web: Beyond the factoid.

Denis Savenkov and Agichtein Eugene. 2016. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, 2016.*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

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

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

Stalin Varanasi and Gnter Neumann. 2015. Question/answer matching for yahoo! answers using a corpus-based extracted ngram-based mapping. In *MLT-Lab DFKI D-66123, 2015*.

Table 4: Examples of Queries and Answers Returned by the System

Question: Bat ran into me, should I be afraid of rabies? **Query**: (afraid) AND (bat OR Chiroptera) AND (Rabies) **Answer**: Bats can carry rabies. Never try and retrieve a sick looking animal. The fecal material from bats can also accumulate and harbor histoplasmosis fungus spores which may cause blindness , and pneumonia like symptoms.Source(s): 23 years in pest control
Question: Is waterproof eyeliner necessary for traveling in hot/humid areas? **Query**: (necessary) AND (travel hot humid areas) AND (waterproof eyeliner) **Answer**: For a softer look, we found that Stila's twist-up Smudge Stick provides the best pencil eyeliner experience, from the flexibility of its application, as it goes on the softest but sets the firmest, to its remarkable staying power.[The Stila Smudge Stick Waterproof liner] was easiest to use of all the eyeliners tested. Once applied and given a few seconds, it sets firmly and was among the hardest to budge for our testers.TheStila Smudge Stick Waterproof Eyelineris an extra-soft, twist-up, thin mechanical pencil that deposits a neat matte line onto the skin.
Question: Can I rent a car in Verona, Italy if I'm 18 years old? **Query**: (18 years old) AND (rent car) AND (Verona Italy) **Answer**: Usually, car rental companies will only allow renters 21 and over. One thing you might check is to see if the company you work for has a corporate account with any car rental companies, where they have contracted a lower minimum age for their employees.Lori J * 8 years ago Thumbs up
Question: Do you need an exotic animal permit to own a peacock? Im asking because im thinking of raising them when i move to my own house also wondering if they do well around chickens **Query**: (exotic animal) AND (peacock) AND (permit) **Answer**: Exotic fowl refers to any avian species that is not indigenous to this state, including ratites (emu, ostrich, rhea, cassowary, etc.There are no state bag or possession limits or closed seasons on exotic animals or fowl on private property. It is against the law to: * Hunt an exotic without a valid hunting license. Thumbs up
Question: Which coffee shops in the netherlands has the best weed? **Query**: (best weed) AND (coffee shops) AND (netherlands) **Answer**: those with tobacco mixed with cannabis, and have made customers smoke in upstairs or downstairs rooms. Notable coffeeshops[edit] * Checkpoint coffeeshop * Mellow Yellow coffeeshop

Di Wang and Eric Nyberg. 2015. Discovering the right answer with clues. In *Proceedings of TREC, 2015*.

Di Wang and Eric Nyberg. 2016. An attentional neural encoder-decoder approach for answer ranking. In *Proceedings of TREC*.