Adaptive Query-based Model for Improved Ranking in Closed Domain Factoid Question Answering

Huey Ling Toh, Lois Wright Hawkes, R. C. Lacher
Department of Computer Science
Florida State University
Tallahassee, Florida 32306, USA.
toh@cs.fsu.edu, lhawkes@fsu.edu, lacher@cs.fsu.edu

Abstract

Closed domain question answering QA systems achieve precision and recall at the cost of complex language processing techniques to parse the answer corpus. We propose a query-based model for indexing answers in a closed domain factoid question answering system. Further, we use a phrase term inference method for improving the ranking order of related questions. We posit that a query can be used as the unique identifier of an answer, and thus, the recognition of a query allows us to retrieve the correct answer. In instances where a query is unrecognized, we infer synonymous relationships with other queries through the use of a user feedback loop to improve the ranking order of closely related questions, where possible. This offers an adaptive, lightweight approach to a factoid question answering system for domain specific knowledge bases with significantly simplified language processing techniques.

1. Introduction

The nature of closed domain question answering (QA) systems opens the discussion for more efficient ways to address the issue of repetitive queries that lead to the same answer. Such occurrences can be observed in questions posed by students to an instructor, or inquiries submitted by consumers about a company's products or services. Course instructors, for instance, would benefit by allowing a QA system to answer similar questions that arise every semester. A QA system that is able to identify related questions and retrieve the corresponding answers with a high degree of precision would greatly assist in providing ubiquitous and round-the-clock access to pertinent course related information.

Recent trends in QA have demonstrated the effectiveness of statistical lexical analysis to extract relevant answers for a given query [1]. In closed domain QA, however, complexity arises as the relatively small answer corpus results in fewer sentences containing the answer. As Molla and Vicedo [2] aptly states, "The haystack of a restricted domain is relatively small, but it also has fewer needles". Therefore, closed domain QAs have to rely on complex language processing techniques, generally performed off-line, to find the answer. Thus, we process the query instead of the answer, retrieving a narrower if not explicit answer set for a given query. We also use human expertise to supply an answer to an unrecognized query (e.g. the course instructor) and provide feedback on the correctly matched query (e.g. the student). User feedback data is assumed to be valid, although every question submitted by a user is presented to the instructor for verification. Our system does not assume to know the answer to every question submitted by the student. Ideally, the system will learn to recognize questions as its knowledge base obtains more input from both the student and the instructor. In summary, we assert that for QA systems operating in a closed domain, there should be a paradigm shift in the approach of the type of information the system should be processing; with input from an authoritative source, the key to success in such systems lies in identifying the right question, and not the answer.

Our prototype system, named Computerized Adaptive Category HElp (CACHE), is an adaptive, lightweight implementation of a closed domain factoid question answering system that uses query-based indexing of the answer corpus. Given that a human expert provides the answers for a specified set of queries in a closed domain system, we propose a method for indexing the answer by identifying the queries that the answers are associated with. In essence, if an answer A to a question Q has already been provided and verified as correct by a human expert, then we postulate that, "A closed domain QA system that recognizes synonymous questions would be able to retrieve the correct answer without the overhead of analyzing the text in the answer corpus." We give the following analogy; the participants in

the renowned television game-show Jeopardy [3] are presented with a set of answers for a given category. Their task is to guess the correct question that corresponds to a given answer where a variety of questions are acceptable for the given answer. We contend that since our system's knowledge base is also limited by its domain, and it already has an existing repository of answers, then its main task is to guess the correct query that corresponds to each answer. In addition, our closed domain setup is built on a premise that each answer is already indexed with an existing question. Thus, even in a sub-optimal scenario, where similar questions are only best guesses, a list of existing related questions (RQs) that resides within the knowledge base may be displayed in some ranking order to the user. User feedback on a correct match between an RQ and Q, may then be used to enhance the accuracy of the system for subsequent questions that are similar.

2. Related work

Researchers of QA systems generally attempt to create computational models that closely mimic human learning, processing and querying activities [4]. The goal of a QA system is to seek and extract specific answers to a user's questions, expressed in the form of natural language. Search engines, a close competitor group to QA systems, perform link analysis and treat queries as unordered sets of words. This approach is successful in processing Boolean type queries and retrieving popular web pages and their related counterparts. The downside, however, is that the user may have to filter through long lists of results to find the correct answers. QA systems, on the other hand, play an important role in supplying precise, timely, and factual answers for time-sensitive queries (e.g. queries submitted by first-responders, security personnel, and etc.) [5], and queries sent through interfaces with limited browsing capabilities and bandwidth (e.g. mobile devices) [6]. In fact, a study by Roussinov et. al [7] reports that open-domain QA systems technology "complements or even rivals keyword-based search engines".

Unlike open domain QA systems, however, closed domain QA systems operate on small corpora, which inherently contribute to their higher precision in filtering out unrelated information. However, closed domain QA systems also have to contend with the challenge of finding the search phrase in the answers, as there are relatively fewer answers to search from [2]. Minock [8] discusses the importance of circumscription, complexity and practicality as the foundations for a good closed domain QA. Circumscription refers to the need for a narrow topic, a level of detail in the coverage of the answers, and the ability to capture and represent factual information. Complexity is concerned with establishing sufficient benefits for a QA system to warrant its implementation. Practicality demands that answers to single-sentenced queries are useful enough to a group of users while simultaneously upholding the feasibility of expending resources required for knowledge acquisition and retrieval.

3. Proposed method

We address the closed domain QA framework by building a proof-of-concept prototype that is restricted to answering questions for a data structures course in computer science. The system is circumscribed due to its limited domain with a distinct ontology, and as such would be a prime platform for answering factoid questions. The criterion for complexity is met as the system attempts to service many sub-topics within the curriculum, and a simple keyword-based search will not return optimal results. Finally, on the issue of practicality, we prescribe an intuitive web interface built on an Ajax [9] platform that is accessible to users over many iterations of the course offering. In addition, our query-based model for indexing answers offers a lightweight approach for a robust closed domain QA.

Our query-centered model redirects the efforts of indexing answers to the questions that they answer. Essentially, our system expends its resources in trying to either recognize a query or locate a list of related queries to the given query (where possible). Notably, CACHE's knowledge base begins as a blank slate as it will learn to recognize closed-domain factoid questions with increased usage through time. CACHE receives input through a web interface from a user who submits a query and a trainer who supplies or updates the answer to a query. The system learns to recognize queries by processing the information exchange between the users and its web interface.

4. Query-based Model for answer indexing

4.1. Query-based answer indexing

Given a many-to-one relationship between the various incarnations of questions that may lead to the same answer, a system that automatically recognizes the synonymous relationship between these questions, will effectively retrieve the correct answer for a given query or increase the efficiency of retrieving related queries ranked by the synonymy between the *phrase terms* of the matched questions. Phrase terms are defined as the words in the query that remain after the removal of stemmed index terms and stop words. While index terms provide clues to the object of the question [10] by the user, phrase terms will help CACHE infer synonymous relationships of the question and existing queries in the knowledge base. This added utility allows CACHE to enhance the ranking order

of the list of retrieved result set. To elaborate, we provide the following simplified example. Suppose the questions

Q1: "What is a binary search tree?"
Q2: "Define a binary search tree",
Q3: "Explain to me about binary search trees."
Q4: "Discuss binary search trees"

have been assigned the same answer A1 by the trainer, then the system must *infer* that the phrase terms "*Define*", "*Explain*", "*Discuss*" are synonymous with the phrase "*What*"). Through observation, if a QA system is able to recognize the pattern between such phrase terms, it can enhance its retrieval process by adapting to the reoccurrences of the pattern, i.e. the more frequent two phrase terms are matched by the users of the system, the greater the synonymous relationship of the two phrase terms. The work of inferring synonymous relationships between phrase terms is encapsulated in the phrase inference module described next.

4.2. Phrase Inference Module

The Phrase Inference (PI) module attempts to locate a synonymous query by employing two distinct thesauruses to establish correlations between phrases in a query. First, the PI module uses a pre-built general thesaurus for the English language to form *static phrase* (SP) clusters. SP clusters are generated based on a real-time lookup of a list of synonymous phrases in the pre-built thesaurus. Next, the PI module maintains an evolving thesaurus, known as the *pseudo phrase-thesaurus* that manages the synonymous relationship between *adaptive phrase* (AP) clusters. AP clusters are formed based on the user's feedback on correctly matched phrases of the submitted user query and the retrieved query.

Term-term correlation is usually determined through the construction of a *keyword connection matrix* [11]. A relationship between two terms t_i and t_j is defined by a normalized correlation factor. In the PI module, a frequent positive user feedback on the matching of questions with phrase terms p_i and p_j would indicate a stronger inferred synonymy between p_i and p_j. The degree of inferred synonymy between p_i and p_j is defined by the *synonymous factor* (SF), which in turn, is computed by a modified version of Ogawa's [11] normalized correlation factor as follows:

$$SF_{i,j} = \frac{m_{i,j}}{n_i + n_j - m_{i,j}}$$

where,

n_i : the number of questions that contain p_i,
n_j : the number of questions that contain p_j, and
$m_{i,j}$: the number of times a question with p_i was correctly matched with a question with p_j.

The SF is updated every time a question with phrase p_i is matched with phrase p_j. Hence, the term AP cluster is used to denote synonymous phrases identified through the pseudo phrase-thesaurus.

In instances where the system is presented with a question Q it does not recognize, the following steps are performed to retrieve the list of closest related queries:

1) Part-of-speech (POS) Tagging. Generate POS tags for each word in Q.
2) Index Term Generation. Generate the index terms that best represent the object of the question. Currently, the PI module identifies and *stems* all the noun phrases (including compound noun phrases) as index terms for a given question.
3) Phrase Term Generation. Generate a phrase term P, defined as the remaining words that are not part of the index terms in Q. Stop words are also removed for efficiency.
4) Question Storage. Tag Q with the list of index terms IT and phrase term P acquired in steps 2 & 3.
5) Phrase inference. Locate a list of synonymous phrase terms SP in the following order of priority: (i) synonymous phrase terms found as static clusters from the built-in thesaurus, and (ii) synonymous phrase terms found in the AP clusters from the pseudo phrase-thesaurus. If the list of SP is empty, then retrieve possible related questions based solely on IT and proceed to step 7.
6) IT and SP combinations. Locate IT and SP combinations in question set. If no matching question sets are found, proceed to Step 8. If matching question sets are found, retrieve a list of all possible related questions RQ, order by (a) maximum length of noun phrases in IT and degree of synonymy of each phrase in SP to P, and (b) minimum length of noun phrases in IT and question frequency. Order (a) is used to prioritize the result set based on the closest match to the desired IT and SP combination, while order (b) is used in a sub-optimal scenario where less specific terms in IT must be used to locate possible matches. Present the list of RQ to user.
7) User feedback. Obtain user feedback and if a related question RQ_i correctly matches the goal of the user's query, the user will provide a positive match feedback. The system uses this feedback to update the AP cluster by tightening the degree of synonymy between P in RQ_i and P in Q.
8) Update query queue. Send Q to the pending query queue. Q is tagged as an unanswered question to be addressed by the trainer who selects the question to be answered and provides the appropriate response. The trainer also has the option of editing existing question and answer sets to provide updates to the information stored in the knowledge base.

5. Experimental evaluation

5.1. Data sets

We use a small training set covering various concepts in the topic of data structures in computer science to simulate the performance of our query-based model for improved answer ranking. We use the mean reciprocal ranking (MRR) [12] metric to determine precision of the ranking order of our result sets. A query that ranked first in the result set

received a reciprocal rank (RR) of 1. A second ranking received the score of 1/2 or 0.5, while the third ranking received a score of 1/3 or 0.33, and so on. A score of 0 is assigned to responses ranked greater than 5. An MRR score closer to 1 is a good indicator of the performance of the system. The ensuing discussion on specific question sets are selected as a representative sample of how our model would enhance the ranking orders of the retrieved result set.

5.2. Results and observations

We begin by looking at the list of available questions in Table 1, that contain index terms that reference the singular noun 'tree'.

Table 1. Questions that contain the index-term 'tree'

No.	Questions	Index terms	Phrase terms
1.	Tell me about rooted trees.	rooted trees, rooted tree, tree, trees	tell me
2.	What is a tree?	trees, tree	what
3.	Explain binary trees	binary trees, binary tree, trees, tree	explain
4.	Describe the difference between binary trees and binary search trees.	difference between binary tree, binary search tree, binary tree, search tree, difference, binary, search, trees, tree	describe
5.	Tell me about the different types of tree traversal methods.	traversal method, traversal, types, tree, trees	tell me

Suppose a user submits the following question,

Q1: Explain the difference binary trees and binary search trees.

Q1 is subsequently analyzed and it is determined that its index terms and phrase term are as follows:

Q1_index-terms: difference between binary tree, binary search tree, binary tree, search tree, difference, binary, search, trees, tree

Q1_phrase: explain

We would like to point out that systems which employ only keyword based searches will return the entire result set in some pre-determined ranking order. Even if such systems operate within the framework of a query-based model, the retrieved result set would still be potentially large or less precise because the occurrence of at least one index term in each question can be found in Q1. CACHE addresses this by improving the ranking order of the result set. A phrase-based evaluation of the phrase term 'explain' is initiated to locate a synonymous phrase term that matches the one found in a question in Table 1. A lookup in the static cluster of the built-in thesaurus determines that the phrase term 'describe' is synonymous to 'explain' and thus reorders the ranking of the result set to:

4. Describe the difference binary trees and binary search trees.
2. What is a tree?
5. Tell me about the types of tree traversal methods.
1. Tell me about rooted trees.
3. Explain binary trees.

Now, the correct synonymous question is ranked first in the result set. The reciprocal ranking (RR) value for Q1 is thus, 1. Next, suppose the user presents the system with the following question,

Q2: What is the difference binary trees and binary search trees?

and the system determines that Q2's index terms and phrase term are as follows:

Q2_index-terms: difference between binary tree, binary search tree, binary tree, search tree, difference, binary, search, trees, tree

Q2_phrase: what

A search in the SP clusters of the built-in thesaurus will fail because there is no known correlation between 'what' and 'describe', 'tell me', or 'explain' in the thesaurus. The system has no choice but to revert to an index term based search and return a sub-optimal result set of questions 1, 3, 4 and 5. The RR value for Q2 is 0.25.

Fortunately, the result set *does* contain the correct question, specifically, question 5. The system takes this opportunity to *learn* from its human trainer by means of the user-feedback loop. When the user identifies question 5 as the closest match to Q2, the PI module forms an AP cluster between the phrase terms 'what' and 'describe'. The higher the frequency of matches of these two phrase terms in the user-feedback loop, the tighter the correlation between the phrase terms in the cluster. In summary, CACHE has *learned* to *infer* that the phrase 'what' is synonymous with 'describe'.

Let us continue by examining the list of questions in Table 2. These questions are conceptually related because their answer sets contain references to the term 'queue'.

Table 2. Questions for answers related to the term 'queue'

No.	Questions	Index terms	Phrase terms
1.	What is the difference between a **stack** and a **queue?**	difference between a stack, stack, queue	what
2.	Explain how **double-ended queues** work.	double-ended queues, queues	explain
3.	Explain the difference between FIFO and LIFO.	difference between fifo, fifo, lifo	explain
4.	Tell me more about FIFO.	fifo	tell me
5.	What is a **stack**?	stack	what
6.	What is a **queue**?	queue	what

Suppose a user submits the following question,

Q3: Describe queues.

Traditional IR systems that derive index-terms from the answer set would establish questions 1, 2, 3, 5, 6 in Table 2 as relevant to a query, although the human user would likely consider questions 3 and 4 not particularly relevant to Q3. CACHE on the other hand, directs its attention at processing the query. We shall now observe how CACHE's learned inference benefits future query-processing efforts.

The system analyzes Q3 and determines that its index terms and phrase term are as follows:

Q3_index-terms: queue, queues
Q3_phrase: describe

The search scope of the system is narrowed down to the list of questions containing the index-terms 'queue' and 'queues'; specifically questions 1, 2, and 6. Nonetheless, a search in the SP clusters of the built-in thesaurus will fail again, because there is no known correlation between the phrase terms 'describe' and 'what', and without the assistance of the PI module, the system would have to resort to an index term based search that returns a sub-optimal result set containing the following questions based on the frequency of the questions being submitted:

1. What is the difference between a stack and a queue?
2. Explain how double-ended queues work.
6. What is a queue?

But with the help of the PI module, the system is able infer the synonymous relationship between the phrase terms 'what' and 'describe' in the AP cluster of the pseudo-phrase thesaurus based on the user feedback for Q2. The system is thus, able to reorder the ranking of the result set as follows:

6. What is a queue?
1. What is the difference between a stack and a queue?
2. Explain how double-ended queues work.

The correct related query is listed first in the result set, giving us a RR value of 1. We would like to add that the adaptive clusters in the pseudo-phrase thesaurus also form transitive relationships between phrase terms. Suppose that after many iterations of the user-feedback loop, correlations are formed for the following pairs of phrase terms in Table 3:

Table 3. Correlations between Phrase terms in Pseudo-phrase Thesaurus

Phrase-term A	Phrase-term B	Synonymous Factor (SF)
what	describe	0.46
what	explain	0.33
explain	show	0.25
discuss	describe	0.42

If a user submits the following question,

Q4: Discuss queues.

and it is determined that the index terms and phrase term for Q4 is as follows:

Q4_index-terms: queue, queues
Q4_phrase: discuss

Then, transitive correlations can be formed by associating the phrase terms 'discuss' with 'describe' and subsequently, 'describe' with 'what'. This will enable the system to produce an optimal ranking for the result set, similar to ranking for Q3, with an RR value of 1. Notably, in instances where there are two competing transitive correlations, the synonymous factor (SF) will be the determining factor in the ranking order of the result set, as phrases that have a higher SF will be given higher priority. The MRR value for all queries Q1, Q2, Q3, and Q4 is therefore, (1+0.25+1+1)/4 = 0.8125, an impressive indicator for the performance of our system using the small training set.

6. Conclusions

While sub-optimal rankings, such as that for Q2 significantly skew the mean, we attribute the behavior to the immaturity of the system; in particular, during the early stages of usage, many low RR values will undermine early MRR values. However, the performance of the system should improve with increased usage through time, as it learns to form correlation between synonymous queries to retrieve an optimally ranked answer set. The system's knowledge base grows as new answers and their corresponding queries are added into the system by the trainer. Although this incurs cost in terms of time and effort by the trainer, we believe that the benefits of prolonged use far outweigh the system's short-term demands on its users. Given its domain-specific nature, the system's knowledge base would grow with consistent training. As the system's knowledge base expands, it should be able to process a user query with little intervention from the trainer.

We have established that it is indeed possible to create a lightweight implementation of a QA system for domain-specific, single-sentenced queries based on a query-centered model. By focusing our resources on processing queries instead of the answer set, lexical analysis, stemming, stop word removal, index term selections, and text categorization procedures are performed solely on the query at run-time. Based on capturing single-sentenced queries, and given the general assumption that the length of a query is relatively shorter than the length of an answer; the pre-processing of queries is therefore significantly less expensive.

As mentioned in the beginning, the CACHE prototype was built as a proof-of-concept endeavor. At its current inception, the system has been tested only with a small training set. We plan to deploy a beta version of the system for further testing to gather additional performance data and address any usability issues that may arise from real-world usage. We believe that improvements can be made to the current user-interface to enhance usability, but a detailed analysis of possible revisions must be reserved until an actual deployment of the system can be executed.

7. References

[1] Levit, M., Hakkani-Tur, D., Tur, G., Gillick, D. 2009. IXIR: A statistical information distillation system. Computer Speech and Language 23 (2009) 527-542.

[2] Molla, D. & Vicedo, J. L. 2006. Question Answering in Restricted Domains: An Overview. Special Section on

Restricted-Domain Question Answering. Association for Computational Linguistics.

[3] Wang, M., Smith, N. Mitamura, T. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 22-32.

[4] Toh, H, Lacher, R.C. 2004. Computerized Adaptive Student Help: Experiments in Low-Cost Artificially Intelligent Automation of Student-Tutor Interaction. WSEAS Transactions on Computer, 1576-1681.

[5] Liu, Y., Narasimhan, N., Vasudevan, V., and Agishtein, E. 2007. Is This Urgent? Exploring Time-Sensitive Information Needs in Collaborative Question Answering. In Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 712-713.

[6] Ortiz-Arroyo, D. 2008. Flexible Question Answering System for Mobile Devices. In Proceedings of 3rd International Conference on Digital Information Management (ICDIM) 2008, 266-271.

[7] Roussinov, D., Fan, W., Robles, J. 2008. Beyond keywords: automated question answering on the web. Communications of the ACM (CACM), September 2008.

[8] Minock, M. 2005. Where are the 'Killer Applications' of Restricted Domain Question Answering? In Proceedings of Knowledge and Reasoning in Question Answering (KRAQ-2005), 98-101.

[9] Paulson, L.D. 2005. Building Rich Web Applications with Ajax. Computer, vol. 38, no. 10, pp 14-17.

[10] Zhang, W., Liu, S., Yu, C., Sun, C., Liu, F., Meng, W. 2007. Recognition and Classification of Noun Phrases in Queries for Effective Retrieval. In Proceedings of the 16th ACM Conference on Information and Knowledge Management. (CIKM 2007), 711-720.

[11] Ogawa, Y., Morita, T., Kobayashi, K. 1991. A Fuzzy Document Retrieval System Using the Keyword Connection Matrix and a Learning Method. Fuzzy Sets and Systems, v. 39, n.2, 163-179.

[12] Voorhees, J., Tice, D.M. 2000. The TREC-8 Question Answering Track Evaluation. Text Retrieval Conference TREC, vol. 8.