Finding Question-Answer Pairs from Online Forums

Gao Cong *§ Long Wang † Chin-Yew Lin‡ Young-In Song ♯ Yueheng Sun †

§Department of Computer Science, Aalborg University, Denmark
†Department of Computer Science and Technology, Tianjin University, Tianjin, China
‡Microsoft Research Asia, Beijing, China
♯ NLP Lab., Department of Computer Science and Engineering, Korea University, Korea
gcong@inf.ed.ac.uk myimilo@gmail.com cyl@microsoft.com song@nlp.korea.ac.kr

ABSTRACT

Online forums contain a huge amount of valuable user generated content. In this paper we address the problem of extracting question-answer pairs from forums. Question-answer pairs extracted from forums can be used to help Question Answering services (e.g. Yahoo! Answers) among other applications. We propose a sequential patterns based classification method to detect questions in a forum thread, and a graph based propagation method to detect answers for questions in the same thread. Experimental results show that our techniques are very promising.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models; I.2.7 [Artificial Intelligence]: Natural Language Processing
General Terms: Algorithms, Experimentation
Keywords: question answering, graph based ranking, labeled sequential patterns, classification, information extraction

1. INTRODUCTION

An online forum is a web application for holding discussions and posting user generated content in a specific domain, such as sports, recreation, techniques, travel etc. Forums contain a huge amount of valuable user generated content on a variety of topics, and it is highly desirable if the human knowledge contained in user generated content in forums can be extracted and reused.

In this paper we focus on mining knowledge in the form of question-answer (QA) pairs from forums. Many forums contain question-answer knowledge. We investigated 40 forums and found that 90% of them contain question-answer knowledge. Mining question-answer pairs from forums has the following applications. First, question-answer pairs are essential to many QA services, including instant answers provided by search engines, QA search systems, and community-based Question Answering (CQA) services. For example, CQA services, such as Yahoo! Answers, Baidu and Naver [1], have recently become very popular. Forum has a longer history than CQA and contains much larger user-generated content. For example, there were about 700,000 questions in the travel category of Yahoo! Answers as of January 2008. We extracted about 3,000,000 travel related questions from six online travel forums. One would expect that a CQA service with large QA data will attract more users to the service, and the question-answer knowledge embedded in forums could greatly enrich the knowledge base of CQA. Second, question-answer pairs seem to be a natural way to improve forum management (including querying and archiving). Questions are usually the focus of forum discussions and a natural means of resolving issues [18]. Access to forum content could be improved by querying question-answer pairs extracted from forums, which highlight the questions asked and the answers given. Third, question-answer knowledge mined from forums can be used to augment the knowledge base of chatbot [7].

Although it is highly valuable and desirable to extract question answer pairs embedded in forums that are largely unstructured, to our surprise none of previous work addresses this problem. Each forum thread usually contains an initiating post and a couple of reply posts. The initiating post usually contains several questions and reply posts may contain answers to the questions in the initiating post or new questions. The asynchronous nature of forum discussion makes it common for multiple participants to pursue multiple questions in parallel.

In this paper, for each forum thread we find questions and their respective answers in the thread. We propose a new method to detect question-answer pairs in forums, which consists of two components, question detection and answer detection.

Question detection. The objective is to detect all the questions within a forum thread. The problem at first glance seems to be easy. Unfortunately, it turns out to be non-trivial on the basis of our analysis on 1,000 questions from forums. Questions in forums are often stated in an informal way and questions are stated in various formats. We found that simple rule based methods, such as question-mark and 5W1H question words, are not adequate for forum data. For example, 30% questions do not end with question marks while 9% sentences ending with question marks are not questions in a forum corpus. We develop a classification-based technique to detect questions using sequential patterns automatically extracted from both questions and non-question sentences in forums as features.

Answer detection. Given the questions detected in a forum thread, we aim to find their answer passages within the same forum thread. Answer detection is difficult due to the following reasons: First, multiple questions and answers may be discussed in parallel and are often interweaved together, while the reply relationship between posts is usually unavailable; Second, one post may contain answers to multiple questions and one question may have multiple replies.

[*]This work was done when Gao Cong worked as a researcher at the Microsoft Research Asia, and Long Wang and Young-In Song were visiting students at the Microsoft Research Asia.

[1]Baidu: http://zhidao.baidu.com; Naver: http://www.naver.com/.

One straightforward approach to finding answer is to cast answer-finding as a traditional document retrieval problem by considering each candidate answer as an isolated document and the question as a query. We then can employ ranking methods, such as cosine similarity, query likelihood language model and KL-divergence language model. However, these methods do not consider the relationship of candidate answers and forum-specific features, such as the distance of a candidate answer from a question.

To model the relationship between candidate answers and make use of forum-specific features, in this paper we develop a new graph-based approach for answer detection. We model the relationship between answers to form a graph using a combination of three factors, the probability assigned by language model of generating one candidate answer from the other candidate answer, the distance of candidate answer from question, and the authority of authors of candidate answer in forums. For each candidate answer, we can compute an initial score of being a true answer using a ranking method. To use the graph to compute a final propagated score, we consider two methods. The first one integrates the initial score after propagation, while the second one integrates the initial score in the process of propagation.

In this paper, we propose and address the problem of extracting question-answer pairs from forums. We make the following main contributions:

First, we develop a classification-based method for question detection by using sequential pattern features automatically extracted from both questions and non-questions in forums.

Second, we propose an unsupervised graph-based approach for ranking candidate answers. Better still, the graph-based approach can be integrated with classification method when training data is available. For example, the results of the graph-based approach can be used as features for supervised method.

Finally, we conduct extensive experiments on several data. The size of our data is much larger than those used in previous work on forums. The main experimental results include 1) our method outperforms rule-based methods for question detection; and 2) the unsupervised graph-based method outperforms other methods including the classification methods [7, 23].

The rest of this paper is organized as follows: The next section discusses related work. Section 3 presents the proposed technique for question detection and Section 4 presents the proposed techniques for answer detection. We evaluate our techniques in Section 5. Section 6 concludes this paper and discusses future work.

2. RELATED WORK

To our knowledge, none of previous work finds question-answer pairs from forum data. There is some research on knowledge acquisition from forums or discussion boards, e.g. [5, 7, 23]. Close to our work is extracting *input-reply* pairs for chatbot knowledge in [23]. The *input-reply* pair is quite different from *question-answer* pair in this paper: First, thread titles are treated as *input* while we detect one or multiple questions from a thread; Second reply post as a whole is considered as a *reply* while one reply post may contain answers to multiple questions. Moreover, a classification model using lexical features and structured features is employed to distinguish reply posts from non-reply posts [23]. In contrast, our graph-based method is unsupervised. Feng et al. [5] implemented a discussion-bot to automatically answer students' queries by matching the reply posts from an annotated corpus of archived threaded discussions with students' query using cosine similarity. The problem there is quite different from ours. The work in [23] summarizes internet relay chat by clustering a chat log into several sub-topics, and a classification method is used to identify responding messages of the first message in each sub-topic. The classification method used in [23, 7] can be adapted to rank candidate answers for answer finding, but experimental results show that our unsupervised techniques achieve better performance than them.

Another related work is extracting question-answer pairs for email summarization [18] that detects interrogative questions using a classification method similar to [21], and build a classifier to find answers using lexical features and email-specific features. In our recent work [4], conditional random field models are trained to extract the contexts and answers of questions from forums and it does not consider question extraction. In contrast, we propose unsupervised techniques for answer detection, and a supervised method using automatically extracted sequence features for question detection. From our experience, it is easy to label questions, but it is much harder and thus expensive to label answers to a question.

Extensive research has been done in TREC or AQUAINT-style question-answering, e.g. [6, 3]. They mainly focus on constructing short answers for a relatively limited types of question, such as factoid questions, from a large document collection. This makes it feasible to classify the answer type and target when a question is parsed. In contrast, our work focuses on the task of locating answers within a forum thread. Typical questions extracted in forums are more complex, and it is difficult to represent and identify answer types for questions of forums, which has long been recognized to be tough [14]. The good news is that we can take advantage of forum-specific features, such as the distance of candidate answer from question, to extract answers from forums.

Research on FAQ retrieval [1, 17] and CQA retrieval [8] has focused on finding similar questions (together with answers) for a user-given question by leveraging answer information. Query expansion [1] and machine translation model [1, 8, 17] are employed to bridge the lexical chasm between question and answer. The above work is complementary to our work, and could be employed to enhance our methods. In this paper we learn the lexical gap between question and answer using query expansion method in [1]. We also notice the work [9] that retrieves question-answer pairs from FAQ pages by leveraging indentation (e.g. table) and line prefix (e.g. Q:, A:). The task there is easier than ours.

The graph-based methods, such as PageRank [2], have shown to be effective in the Web search, where explicit links between web documents are present. There is also some work building graphs induced by implicit relationships between documents [15, 12, 11]. A graphical model[11] is employed to estimate the joint probability of all answers for answer processing in QA. A graph built using language models is leveraged for document retrieval in [12]. Somewhat more closely related to our work is the LexRank [15] for re-ranking candidate answers. LexRank method uses cosine similarity between candidate answers to build a undirected graph while we build weighted directed graph. In addition, we adopt a different propagation strategy to compute ranking scores.

3. ALGORITHMS FOR QUESTION DETECTION

For question detection in forums, rules, such as question mark and 5W1H words, are not adequate. With question mark as an example, we find that 30% questions do not end with question marks while 9% sentences ending with question marks are not questions in a corpus. This is because 1) questions can be expressed by imperative sentences, e.g. "*I am wondering where I can buy cheap and good clothing in beijing.*"; 2) question marks are often omitted in forums; 3) short informal expressions, such as "really?", should not be regarded as questions. To complement the inadequacy of simple rules, in this paper we extract labeled sequential patterns from both questions and non-questions to characterize them, and then use the discovered patterns as features to build classifiers for question detection. Labeled sequential patterns are used to identify comparative sentences [10] and erroneous sentences [19].

We next first explain labeled sequential patterns (LSPs) and present how to use them for question detection. Consider a question, "*i want to buy an office software and wonder which software company is best.*" "*wonder which...is*" would be a good pattern to characterize the question.

A labeled sequential pattern (LSP), p, is an implication in the form of $\texttt{LHS} \rightarrow \texttt{c}$, where LHS is a sequence and c is a class label. Let I be a set of items and L be a set of class labels. Let D be a sequence database in which each tuple is composed of a list of items in I and a class label in L. We say that a sequence $s_1 =< a_1, ..., a_m >$ is *contained* in a sequence $s_2 =< b_1, ..., b_n >$ if 1) there exist integers $i_1, ...i_m$ such that $1 \leq i_1 < i_2 < ... < i_m \leq n$ and $a_j = b_{i_j}$ for all $j \in 1, ..., m$, and 2) the distance between the two adjacent items b_{i_j} and $b_{i_{j+1}}$ in s_2 needs to be less than a threshold λ (we used 5). Similarly, we say that a LSP p_1 is contained by p_2 if the sequence $p_1.LHS$ is contained by $p_2.LHS$ and $p_1.c = p_2.c$. Note that it is not required that s_1 appears continuously in s_2.

The *support* of p, denoted by $\texttt{sup}(p)$, is the percentage of tuples in database D that contain the LSP p. The probability of the LSP p being true is referred to as "the confidence of p ", denoted by $\texttt{conf}(p)$, and is computed as $\frac{\texttt{sup}(p)}{\texttt{sup}(p.LHS)}$. The support is to measure the generality of the pattern p and minimum confidence is a statement of predictive ability of p. For example, consider a sequence database containing three tuples $t_1 = (< a, d, e, f >, Q)$, $t_2 = (< a, f, e, f >, Q)$ and $t_3 = (< d, a, f >, NQ)$. One example LSP $p_1 = < a, e, f > \rightarrow Q$, which is contained in tuples t_1 and t_2. Its support is 66.7% and its confidence is 100%. As another example, LSP $p_2 = < a, f > \rightarrow Q$ with support 66.7% and confidence 66.7%. p_1 is a better indication of class Q than p_2.

To mine LSPs, we need to pre-process each sentence by applying Part-Of-Speech (POS) tagger MXPOST Toolkit[2] to tag each sentence while keeping keywords including 5W1H, modal words, "wonder", "any" etc. For example, the sentence "*where can you find a job*" is converted into "*where can PRP VB DT NN*", where "*PRP*", "*VB*", "*DT*" and "*NN*" are POS tags. Each processed sentence becomes a database tuple. Note that the keywords are usually good indications of questions while POS tags can reduce the sparseness of words. The combination of POS tags and keywords allows us to capture representative features for question sentences by mining LSPs. Some example LSPs include "<*anyone, VB, how*> $\rightarrow$ *Q*", and "<*what, do, PRP, VB*> $\rightarrow$ *Q*". Note that the confidences of the discovered LSPs are not necessary 100%, their lengths are flexible and they can be composed of contiguous or distant words/tags.

Given a collection of processed data, we will mine LSPs by imposing both *minimum support* threshold and *minimum confidence* threshold. The *minimum support* threshold is to ensure that the discovered patterns are general while the *minimum confidence* threshold ensures that all discovered LSPs are discriminating and are capable of predicting question or non-question sentences. In our experiments, we empirically set *minimum support* at 0.5% and minimum confidence at 85% [3]. Existing frequent sequential pattern mining algorithms (e.g. [16]) do not consider minimum confidence constraint, and we adapt them to mining LSPs with constraints.

[2]http://www.cogsci.ed.ac.uk/~jamesc/taggers/MXPOST.html

[3]The thresholds were set to get thousands of patterns, and we did not try to search thresholds to optimize classification results in experiments

Each discovered LSP forms a binary feature as the input for a classification model. If a sentence includes a LSP, the corresponding feature is set at 1. We build a classifier to detect questions using Ripper classification algorithm.

4. ALGORITHMS FOR ANSWER DETECTION

In this section, we present our techniques to find answers in forums for extracted questions. The input is a forum thread with the questions annotated; the output is a list of ranked candidate answers for each question. We observed that paragraphs are usually good answer segments in forums. For example, given a question "*Can anyone tell me where to go at night in Orlando?*", its answer "*You would be better off outside the city. look into International drive or Lake Buena Vista. for nightlife try Westside in the Disney Village. have a look at MARRIOTTVILLAGE.COM. located in LBV*" is a paragraph. We also observed that the answers to a question usually appear in the posts after the post containing the question. Hence for each question we assume its set of candidate answers to be the paragraphs in the following posts of the question. We would point out that the proposed techniques are equally applicable to other kinds of segments.

4.1 Preliminary

We will briefly introduce three IR methods to rank candidate answers for a given forum question, and then discuss how to adapt the classification method [23, 7] to rank answers.

Cosine Similarity. Given a question $\mathbf{q}$ and a candidate answer $\mathbf{a}$, their cosine similarity weighted by inverse document frequency (idf) can be computed as follows:

$$\texttt{COS}(\mathbf{q}, \mathbf{a}) = \frac{\sum_{w \in \mathbf{q}, \mathbf{a}} f(w, \mathbf{q}) \times f(w, \mathbf{a})(idf_w)^2}{\sqrt{\sum_{w \in \mathbf{q}} (f(w, \mathbf{q}) idf_w)^2} \times \sqrt{\sum_{w \in \mathbf{a}} (f(w, \mathbf{a}) idf_w)^2}} \tag{1}$$

where $f(w, X)$ is the frequency of word w in X, idf_w is inverse document frequency (idf) (each document corresponds to a post in the thread of question $\mathbf{q}$).

Query likelihood language model. The probability of generating a question $\mathbf{q}$ from language models of candidate answers can be used to rank candidate answers. Given a question $\mathbf{q}$ and a candidate answer $\mathbf{a}$, the ranking function for the *Query likelihood language model* using Dirichlet smoothing is as follows:

$$\texttt{QL}(\mathbf{q}|\mathbf{a}) = \prod_{w \in \mathbf{q}} P(w|\mathbf{a}) \tag{2}$$

$$P(w|\mathbf{a}) = \frac{|\mathbf{a}|}{|\mathbf{a}| + \lambda} \times \frac{f(w, \mathbf{a})}{|\mathbf{a}|} + \frac{\lambda}{|\mathbf{a}| + \lambda} \times \frac{f(w, C)}{|C|} \tag{3}$$

where $f(w, X)$ denotes the frequency of word x in X, and C is the background collection used to smooth language model.

KL-divergence language model. It has been shown that a model comparison approach outperforms query-likelihood model for information retrieval [13]. We construct unigram question language model $M_{\mathbf{q}}$ for question $\mathbf{q}$ and unigram answer language model $M_{\mathbf{a}}$ for answer candidate answer $\mathbf{a}$. We then compute KL-divergence between the answer language $M_{\mathbf{a}}$ and question language model $M_{\mathbf{q}}$ below.

$$\texttt{KL}(M_a||M_q) = \sum_w p(w|M_a) log(p(w|M_a)/p(w|M_q)) \tag{4}$$

Classification based re-ranking. Recall that as discussed in related work, classification method [23, 7] is employed to extract knowledge from forums, though not question-answer pairs. Classifiers are built to extract *input-response* pairs [7] using content features (e.g. *the number overlapping words between input and reply post*) and structural features (e.g. *is the reply posted by the thread starter*). The other work [23] uses slightly different features. We can treat each question and candidate answer pair as an instance, compute features for the pair, and train a classifier. The value returned by a classifier, called as *classification scores*, can be used to rank the candidate answers of a question. Note that classification based re-ranking method needs training data which are usually expensive to get.

The methods presented above do not make use of any inter candidate answer information, while the candidate answers for a questions are not independent in forums. We next present an unsupervised graph-based method that considers the inter-relationships of candidate answers.

4.2 Graph based propagation method

The graph-based propagation method has been successful in Web search where links are usually obvious. However, it remains largely unexplored to apply graph-based propagation for answer finding especially in forum data. Intuitively, if a candidate answer is related to (e.g. similar to) an authoritative candidate answer with high score, the candidate answer, which may not have a high score, is also likely to be an answer. In this section, we first present how to build graphs for candidate answers, and then how to compute ranking scores of candidate answers using the graph.

4.2.1 Building Graphs

Given a question $\mathbf{q}$, and the set $\mathbf{A_q}$ of its candidate answers, we build a weighted directed graph denoted as (V, E) with weight function $w : E \rightarrow \Re$, where V is the set of vertices and E is the set of directed edges and $w(u \rightarrow v)$ is the weight associated with edge $u \rightarrow v$. Each candidate answer in $\mathbf{A_q}$ will correspond to a vertice in V. The problem is how to generate the edge set E.

Given two candidate answers $\mathbf{a}_o$ and $\mathbf{a}_g$, we use KL-divergence language model $\mathtt{KL}(\mathbf{a}_o|\mathbf{a}_g)$ (resp. $\mathtt{KL}(\mathbf{a}_g|\mathbf{a}_o)$) to determine whether there will be an edge $\mathbf{a}_o \rightarrow \mathbf{a}_g$ (resp. $\mathbf{a}_g \rightarrow \mathbf{a}_o$). The use of KL divergence language model can be motivated by the following example: consider two candidate answers for a question $\mathbf{q}$: *can tell me some about hotel*. $\mathbf{a}_1$: *world hotel is good but i prefer century hotel* and $\mathbf{a}_2$: *world hotel has a very good restaurant*. Knowing that $\mathbf{a}_2$ is answer would provide evidence that $\mathbf{a}_1$ is also somewhat important and could be answer, but not vice versa. This is because $\mathbf{a}_1$ concerns both *world hotel* and *century hotel* while $\mathbf{a}_2$ concerns only *world hotel*. KL-divergence language model allows us to capture the asymmetry in how the authority is propagated. The cosine similarity used in [15] to construct a graph cannot capture the asymmetry.

We next introduce the definitions of *generator* and *offspring* that will frame edge generation.

Definition 1: Given two candidate answers $\mathbf{a}_o$ and $\mathbf{a}_g$, if $1/(1 + \mathtt{KL}(\mathbf{a}_o|\mathbf{a}_g))$ is larger than a given threshold θ, an edge will be formed from $\mathbf{a}_o$ to $\mathbf{a}_g$. We say that $\mathbf{a}_g$ is a *generator* of $\mathbf{a}_o$ and $\mathbf{a}_o$ is an *offspring* of $\mathbf{a}_g$. □

According to the definition, we can determine whether to generate an edge from $\mathbf{a}_o$ to $\mathbf{a}_g$, and similarly we can determine the presence of an edge from $\mathbf{a}_o$ to $\mathbf{a}_g$ by comparing $\mathtt{KL}(\mathbf{a}_g|\mathbf{a}_o)$ and θ. The parameter θ in the definition is determined empirically and we found in our experiments that our methods are not sensitive to the parameter. We allow self-loop, i.e., each candidate answer can be its own generator. The self-loop edge will allow that one candidate answer is its own generator and offspring. This will also function as a smoothing factor in computing weight and authority. Note that one candidate answer can be a generator of multiple candidate answers and that it is possible for one candidate answer to have no generator. In the extreme case, there is no edges in the graph and thus graph propagation is turned off.

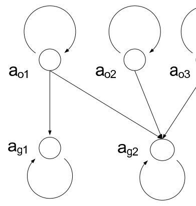

Figure 1: Example of graph built from candidate answers

After we have both vertices and edges, the remaining problem is to compute weight for each edge. One straightforward way is to use the KL-divergence score. To achieve better performance, we will consider two more factors in computing weight.

First, we observe that the replying posts far away from the question post usually are less likely to contain answers for the questions in the post in forums. Hence, when building the digraph for a question, we consider the distance between a candidate answer and the question, denoted by $\mathtt{d}(\mathbf{q}, \mathbf{a})$.

Second, we observe that in forums the posts from authors with high authority are more likely to contain answers. Some forums may provide the authority level of authors while many forums do not have the information. We estimate the authority of an author in terms of the number of his replying posts and the number of threads initiated by him: $\mathtt{author}(\mathrm{i}) = \frac{(\#reply_i)^2/\#start_i}{\max_{j \in I}((\#reply_j)^2/\#start_j)}$, where I is the set of all authors in a forum.

Given two candidate answers $\mathbf{a}_o$ and $\mathbf{a}_g$, the weight for edge $\mathbf{a}_o \rightarrow \mathbf{a}_g$ is computed by a linear interpolation of the three factors, namely the similarity computed from KL-divergence $\mathtt{KL}(\mathbf{a}_o|\mathbf{a}_g)$, the distance of $\mathbf{a}_g$ from $\mathbf{q}$, and the authority of the author of $\mathbf{a}_g$.

$$w(\mathbf{a}_o \rightarrow \mathbf{a}_g) = \frac{1}{1 + \mathtt{KL}(P(\mathbf{a}_o)|P(\mathbf{a}_g))} + \lambda_1 \frac{1}{\mathtt{d}(\mathbf{a}_g, \mathbf{q})} \quad (5)$$
$$+\lambda_2 \mathtt{author}(\mathbf{a}_g)$$

We employ the normalization method in PageRank algorithm [2] to normalize weight. Intuitively, given a "damping factor" λ [4] [2] (was set at 0.01), a candidate answer $\mathbf{a}_o$ and a set of its generators $G_{\mathbf{a}_o}$ in the set of candidate answers A, we normalized the weight $w(a_o \rightarrow a_g)$ among all generators g of $\mathbf{a}_o$, $g \in G_{\mathbf{a}_o}$.

$$nw(a_o \rightarrow a_g) = \lambda \frac{1}{|\mathbf{A_q}|} + (1 - \lambda) \frac{w(\mathbf{a}_o \rightarrow \mathbf{a}_g)}{\sum_{g \in G_{\mathbf{a}_o}} w(\mathbf{a}_o \rightarrow g)}. \quad (6)$$

If a candidate answer has multiple generators, the importance of the weight of the generators will be normalized across its generators. We next illustrate the normalization with an example. Consider the graph built from the candidate answers of a question given in Figure 1. The candidate answer a_{o_1} has three generators, a_{g_1}, a_{g_2} and itself. The weight of edge $a_{o_1} \rightarrow a_{g_1}$ will be normalized from three weights $w(a_{o_1} \rightarrow a_{g_1})$, $w(a_{o_1} \rightarrow a_{g_2})$ and $w(a_{o_1} \rightarrow a_{o_1})$. Note that a candidate answer can be a generator of itself and would function as a smoothing factor.

4.2.2 Computing Propagated Scores

We develop two approaches to integrating the propagated authority with the initial ranking scores that are computed using any approach described in Section 4.1.

[4]To make sure that the Markov chains are always irreducible and aperiodic

Propagation without initial score: For each candidate answer $a \in C_a$, the methods introduced in Section 4.1 can be employed to compute its initial ranking score. Moreover, we also compute its authority value, which can be understood as the "prior" of the candidate answer to be used to adjust the initial ranking score. The product of the authority value and the initial ranking score between candidate answer $\mathbf{a}$ and question $\mathbf{q}$ will be returned as the final ranking score for $\mathbf{a}$.

$$Pr(\mathbf{q}|\mathbf{a}) := authority(\mathbf{a}) \times score(\mathbf{q}, \mathbf{a}) \tag{7}$$

where $score(\mathbf{q}, \mathbf{a})$ is the initial ranking score, and $authority(\mathbf{a})$ implies the significance of answer $\mathbf{a}$ in the answer graph.

We next describe how to compute the authority score for a candidate answer $\mathbf{a}$. Inspired by the work in [12] that computes the authority of documents in information retrieval, we can compute authority for a candidate answer $\mathbf{a}$ by the weighted in-degree for each candidate answer $\mathbf{a} \in C_a$ in the given graph, i.e. the initial authority of $\mathbf{a}_g$,

$$authority(\mathbf{a}_g) = \sum_{\mathbf{a}_o \in C_a} nw(\mathbf{a}_o \rightarrow \mathbf{a}_g). \tag{8}$$

As observed in [12], if the authority of offspring $\mathbf{a}_o$ (generated by $\mathbf{a}_g$) of $\mathbf{a}_g$ is low, the authority of $\mathbf{a}_g$ would not be high. Intuitively, if all answers generated by a specific answer are not central, it will not be central. Note that the reverse may not be true: even if the generator of $\mathbf{a}_g$ is important, it is not necessary that its offspring $\mathbf{a}_o$ is important. The motivation can be modeled by defining the authority of $\mathbf{a}_g$ recursively as follows:

$$authority(a_g) = \sum_{a_o \in C_a} nw(a_o \rightarrow a_g) \times authority(a_o) \tag{9}$$

The authority propagation will converge. The edge weights after normalization in Equation 6 correspond to transition probabilities for a Markov chain that is aperiodic and irreducible, and converges to the stationary distribution regardless of where it begins. The stationary distribution of a Markov chain can be computed by a simple iterative algorithm called power method which converged very quickly in our experiments.

Propagation with initial score: Unlike the first approach, this approach incorporates the initial score between candidate answer and question into propagation. This propagation mechanism is inspired by the work [15]. Given a question $\mathbf{q}$ and its set $C_\mathbf{q}$ of candidate answer, the ranking score of a candidate answer $\mathbf{a}$, $\mathbf{a} \in C_\mathbf{q}$ will be computed recursively as follows.

$$Pr(\mathbf{q}|\mathbf{a}) = \lambda \frac{Pr(\mathbf{q}|\mathbf{a})}{\sum_{\mathbf{t} \in C_\mathbf{q}} Pr(\mathbf{q}|\mathbf{t})} + (1-\lambda) \sum_{\mathbf{v} \in C_\mathbf{q}} nw(\mathbf{v} \rightarrow \mathbf{a}) \times Pr(\mathbf{q}|\mathbf{v}) \tag{10}$$

where the parameter λ is a trade-off between the score of $\mathbf{a}$ and the scores of $\mathbf{a}$'s offsprings in the equation, and is determined empirically. For higher value of λ, we give more importance to the score of the candidate answers itself compared to the score of its offsprings. The weight nw is computed in Equation 6 .

The propagation will converge and the stationary distribution of a Markov chain can be computed by an iterative power method algorithm. The denominators $\sum_{t \in C_\mathbf{q}} Pr(\mathbf{q}|\mathbf{t})$ are used for normalization and the second term in the equation is also normalized so that the weights of all edge leading out of any candidate answer will sum up to 1. Therefore, they can be treated as transition probabilities. With probability (1-λ), a transition is made to the nodes that are generators of the current node. Every transition is weighted according to the similarity distributions.

4.3 Integration with other methods

One benefit of graph-based method is that it is complementary with supervised methods for knowledge extraction, e.g. [23, 7] and techniques for question answering, e.g. [1, 22, 17]. This section will discuss them respectively. First, the graph-based model can be integrated with classification model when training data is available. Second, we learn lexical matchings between questions and answers to enhance the IR methods for answer ranking, and thus graph-based methods.

The integration of graph-based model and classification model. Graph-based method and classification method can be integrated in two ways when training data is available. First, for each candidate answer and question pair, the results returned by graph-based methods can be added as features for classification method to determine if the candidate answer is an answer of the question. The returned classification score for each candidate answer will be used to rank all the candidate answers of a question. In doing so, the classification model can make use of the relationship between candidate answers. Second, the classification score returned by a classifier is often (or can be transformed into) the probability for a candidate answer being a true answer and can be used as initial score for propagation of graph-based model.

Bridging the lexical chasm between questions and answers for graph-based model. Question and answer may use different words. For example, `why` $\rightarrow$ *because*. This is studied in previous work e.g. [1, 8, 17]. The benefit from enhancing question with answer words can also be compared with that from topic models in TREC question answering [22]. However, it is difficult to build a variety of topic models for different questions as mentioned in [22]. We learn the mapping by computing the mutual information between question terms and answer terms in a training set of QA pairs [5]. We then make use of the answer terms by adding the top-k terms with the highest mutual information to expand question as in [1]. Interested readers can refer to [1] for details.

5. EXPERIMENTS

In this section, we will evaluate the techniques for question detection and answer detection.

Data. We selected three forums of different scales to obtain source data. 1) We obtained 1,212,153 threads from TripAdvisor forum [6]; 2) We obtained 86,772 threads from LonelyPlanet forum [7]; 3) We obtained 25,298 threads from BootsnAll Network [8].

From the source data, we generated two datasets for question identification. From the TripAdvisor data, we randomly sampled 650 threads. Each thread in our corpus contains at least two posts and on average each thread consists of 4.46 posts. Two annotators were asked to tag questions and their answers in each thread. The kappa statistic for identifying questions is 0.96. The kappa statistic for linking answers and questions given a question is 0.69, which is lower than that for questions. The reason would be that questions are easier to annotate while it is more difficult to link answers with questions. We then generated two datasets by taking the union of the two annotated data, denoted as `Q-TUnion`, and the intersection, denoted as `Q-TInter`. In `Q-TUnion` a sentence was labeled as a question if it was marked as a question by either annotator; In `Q-TInter` a sentence was labeled as a question if both annotators marked it as a question.

[5]We extracted 300,000 question-answer pairs from Yahoo! Answers as training set.

[6]http://www.tripadvisor.com/ForumHome

[7]http://www.lonelyplanet.com/thorntree/index.jspa

[8]http://boards.bootsnall.com/eve/ubb.x

Data	Method	Prec(%)	Rec(%)	F_1(%)
Q-TUnion	5W-1H words	69.0	14.8	24.4
	Question Mark	96.8	78.4	86.6
	SM [18]	81.9	87.8	84.6
	Our	96.5	98.5	97.5
Q-TInter	5W-1H words	69.0	15.3	25.0
	Question Mark	98.7	77.6	86.9
	SM [18]	92.7	86.8	89.7
	Our	97.8	97.0	97.4

Table 1: Performance of Question Detection

From the source data, we generated five datasets for answer detection. First, two datasets are generated from the 650 annotated threads by taking the union and intersection of the two annotated data, denoted as `A-TUnion` and `A-TInter`, respectively. An answer candidate was labeled as an answer if either annotator marked it as an answer for `A-TUnion`, and if both annotators marked it for `A-TInter`. Here we used questions in `Q-TInter`. Second, we randomly sampled 100 threads from TripAdvisor, LonelyPlanet and BootsnAll, respectively. Thus we get another three datasets, denoted as `A-Trip2`, `A-Lonely` and `A-Boots`.

5.1 Evaluation on question detection

The experiment is to evaluate the performance of question detection method against simple rules and the method in [18], denoted as `SM`. We randomly selected two subsets (containing 1221 sentences) from `Q-TUnion` and `Q-TInter`, respectively, as test date, and randomly selected another 4003 sentences from the remaining sentences in `Q-TUnion` and `Q-TInter`, respectively, to mine LSPs. Table 1 gives the results of Precision, Recall and F_1-score. The experimental results are obtained through 10-fold cross-validation for `SM` and our method. The rule `5W-1H words` is that a sentence is a question if it begins with 5W-1H words; The rule `Question Mark` is that a sentence is a question if it ends with question mark. Although `Question Mark` achieves good precision, its recall is low. Our method outperforms the simple rule approaches and also `SM`, and the improvements are statistically significant (p-value $<$ 0.001). The main reason for the improvement could be that the discovered labeled sequential patterns are able to characterize questions. Note that both SM and our approach employ Ripper to build classifiers. For example, in one experiment on `Q-TUnion`, we mined 2,316 patterns for questions, which consist of the combination of question mark, keywords (e.g. 5W1H words) and POS tags (e.g. 1,074 patterns contain question mark); we also mined 2,789 patterns for non-questions.

5.2 Evaluation on answer identification

In this subsection, we evaluate the performance of graph-based answer detection method and compare it with other methods. We also study the performance of integrating graph-based method and classification method, and the effectiveness of question-answer lexical mapping.

Metrics. We evaluated the performance of our approaches for answer finding using three metrics - Mean Reciprocal Rank (MRR) [20], Mean Average Precision (MAP) and Precision@1(P@1). MRR is the mean of the reciprocal ranks of the first correct answers over a set of questions. This measure gives us an idea of how far down we must look in the ranked list in order to find a correct answer. MAP is the mean of the average of precisions computed after truncating the list after each of the correct answers in turn over a set of questions. MRR considers the first correct answer while MAP considers all correct answers. P@1 is the fraction of the top-1 candidate answers retrieved that are correct. In the context of extracting question-answer pairs, we are usually more interested in the top-1 returned answer and thus the P@1 measure would be ideal. However, some types of questions, such as asking for advice, often have more than 1 correct answer and it would be useful to find alternative answers. Hence, we report results using all the three metrics.

Method	Abbrev.
Nearest Answer/Random Guess	NA
LexRank [15]	Lex
Classification [7, 23] (Section 4.1)	Cla
Cosine similarity (Sec. 4.1)	CS
Query Likelihood language model (Sec. 4.1)	QL
KL divergence language model (Sec. 4.1)	KL
Graph+Cosine similarity (Sec. 4.2)	G+CS
Graph+Query Likelihood language model (Sec. 4.2)	G+QL
Graph+KL divergence language model (Sec. 4.2)	G+KL
Graph(Classification) (Sec. 4.3)	G(Cla)
Classification(Graph) (Sec. 4.3)	Cla(G)

Table 2: The methods and their abbreviations

Method	All questions			Question with answer		
	P@1(#)	MRR	MAP	P@1	MRR	MAP
NA	0.525(806)	0.585	0.504	0.644	0.718	0.618
Lex	0.529(812)	0.616	0.588	0.649	0.756	0.721
Cla	0.588(903)	0.667	0.631	0.722	0.818	0.774
CS	0.559(858)	0.643	0.601	0.686	0.789	0.737
QL	0.568(872)	0.644	0.586	0.697	0.791	0.719
KL	0.578(887)	0.659	0.621	0.709	0.809	0.762
G+CS	0.603(925)	0.677	0.639	0.739	0.830	0.784
G+QL	0.620(952)	0.687	0.632	0.761	0.843	0.775
G+KL	**0.665**(1,021)	**0.719**	**0.686**	**0.816**	**0.882**	**0.842**

Table 3: Results on `A-TUnion` data

Methods. Table 2 lists the methods evaluated and their abbreviations. The better of *the Nearest Answer* and *Random Guess* was reported as a baseline. The LexRank algorithm [15] was used for answer finding. Although LexRank assumed sentences as answer segments in [15], it is applicable to paragraphs used in our experiments. The classification methods used in [7, 23] were adapted for re-ranking candidate answers (Section 4.1) and the better one was reported. Graph+Cosine similarity(`G+CS`) (resp. `G+QL` and `G+KL`) represents the graph-based model using cosine similarity (resp. Query Likelihood and KL divergence) as the initial ranking score. Graph(Classification) represents to use results of the classification based re-ranking [7, 23] as the initial score and Classification(Graph) represents to use the results of graph-based models as features for classification based re-ranking.

5.2.1 Results on fully annotated dataset

Table 3 shows the P@1(together with the number of correct top-1 answers), MRR scores and MAP scores on `A-TUnion` data containing 1,535 questions from 600 threads [9]. Each question has 10.5 candidate answers on average. As shown in Table 3, graph-based methods significantly outperform their respective counterparts in terms of all the three measures as expected. For example on `A-TUnion` data `G+KL` performs 15.1% (resp. 15.7%) better than `KL` on all questions (resp. questions with answers) in terms of P@1 and the improvements are statistical significant (*p-value* $<$ 0.001). The main reason for the improvements is that `G+KL` takes advantage of the relationship of candidate answers and some forum-specific features. The reason for reporting the results on the set of questions with answers is that 284 questions do not have answers and setting thresholds for the methods in Table 2 failed to de-

[9] We observed qualitatively similar results on `A-TInter`. Due to space limitation, we will not report the experimental results on `A-TInter`.

Method	All questions		
	P@1(#)	MRR	MAP
NA	0.735(357)	0.800	0.754
Lex	0.755(367)	0.830	0.794
Cla	0.803(390)	0.865	0.832
CS	0.772(375)	0.847	0.798
QL	0.796(387)	0.859	0.806
KL	0.778(378)	0.849	0.818
G+CS	0.823(400)	0.879	0.838
G+QL	0.854(415)	0.895	0.849
G+KL	**0.877**(426)	**0.909**	**0.880**

Table 4: Results on first question subset of `A-TUnion` data

tect the questions without answers (deteriorated performance), i.e. all the methods identified wrong answers for all the 284 questions. Therefore, the results reported on questions with answers would be more informative to compare the performance of these methods. We will further discuss detecting questions without answers. Note that the parameters of graph-based method were determined on a development set with 50 threads (to be explained later).

We also observed that `G+KL` outperforms `G+QL` and `G+CS` and they all outperform the baseline method `NA`. The improvements are statistically significant on all three metrics (p-value < 0.001). The *nearest answer* method performed better than *random guess* method and we only report the former. It is interesting to observe that `G+KL` outperforms the supervised methods adapted from [23, 7] (statistically significant, p-value < 0.001). The classification results are reported on the average of 10-fold cross-validation on 5 runs (20-fold cross-validation returned similar results). The reason for the superiority of `G+KL` is that it leverages the relationship between candidate answers while the supervised model [23, 7] does not. `G+KL` also significantly outperforms Algorithm `Lex`.

Improved results on subsets. As we have seen, our approaches work well on questions with answers. However, the overall performance deteriorated due to the questions without answers. Hence, the performance should be improved if we can successfully detect the questions without answers. Unfortunately, it is a tough problem, and setting thresholds did not work as mentioned earlier. We observed that most of first questions of each thread have answers. Of 486 first questions, only 21 of them do not have answers for `A-TUnion` data and 45 for `A-TInter` data. The results on the subset of `A-TUnion` are given in Table 4. Due to space limitation, we do not give results on subset of `A-TInter`. The table shows that the performance on the subset is much better than that on all the questions, although the subset contains only one third of all question-answer pairs in forums. In real QA services, correct answers would be desirable for users' satisfaction.

In addition, the classification methods [7] would tell if a candidate answer is a real answer to a question, and thus we can determine if a question has answers by checking each pair of question and answer candidate. However the result was very poor. Instead, we built a classifier by treating each question and all its candidate answers as an instance. In addition to similarity features between question and its candidate answers, we extracted question-specific features, such as location of questions in a thread. The classifier returned 689 questions of which 49 do not have answers. We did error analysis and found neither similarity features nor question-specific features are very effective. For example, only 17.8% of the questions without answers do not have common words with candidate answers while 15.6% of questions with answers do not, either. This is still an open problem to be investigated in the future.

Graph-based propagation methods. The objective of this experiment is to evaluate the different options in graph-base propagation methods. The options include:

Method	All questions			Question with answer		
	P@1(#)	MRR	MAP	P@1	MRR	MAP
$G_{K,1}$+CS	0.563(864)	0.651	0.618	0.691	0.799	0.758
$G_{K,1}$+QL	0.584(897)	0.657	0.596	0.717	0.807	0.732
$G_{K,1}$+KL	0.626(961)	0.688	0.648	0.768	0.845	0.795
$G_{A,1}$+CS	0.603(925)	0.677	0.639	0.739	0.830	0.784
$G_{A,1}$+QL	0.620(952)	0.687	0.632	0.761	0.843	0.775
$G_{A,1}$+KL	**0.665**(1,021)	**0.719**	**0.686**	**0.816**	**0.882**	**0.842**
$G_{K,2}$+CS	0.541(831)	0.622	0.598	0.664	0.763	0.733
$G_{K,2}$+QL	0.546(838)	0.624	0.600	0.670	0.766	0.736
$G_{K,2}$+KL	0.546(838)	0.625	0.600	0.670	0.767	0.736
$G_{A,2}$+CS	0.616(946)	0.683	0.656	0.756	0.839	0.805
$G_{A,2}$+QL	0.623(956)	0.688	0.660	0.764	0.844	0.810
$G_{A,2}$+KL	0.625(959)	0.689	0.661	0.767	0.846	0.812

Table 5: The evaluation of graph-based method on `A-TUnion`

- Two propagation methods. *Propagation without initial score* (by default and denoted as G_1) and *Propagation with initial score* (denoted as G_2);
- Different ranking methods including `CS`, `QL` and `KL`
- Different methods of computing weight. We would like to know the usefulness of distance and authority in computing weight. Hence, we compare using KL-divergence alone, denoted as G_K, and using all the three factors as in Equation 5 (by default and denoted as G_A).

Note that in graph-based method, *propagation without initial score* method and all the three factors in Equation 5 are used by default. For example, `G+KL` represents $G_{A,1}$+KL. By combining the different options, we got several methods shown in Table 5. For example $G_{K,2}$+KL represents to use the propagation method, *propagation with initial score* and use `KL` to compute weight. The performance of using Equation 5, G_A, always outperforms using KL divergence alone G_K. This demonstrates the usefulness of forum-specific features used in Equation 5. The ranking method `KL` always performs better than other two methods `CS` and `QL`. We also found that *propagation without initial score* G_1 often outperforms the other G_2, but not always.

There are three parameters in the graph-based model. They are determined on a development set of 157 questions from 50 threads by considering P@1 in `G+KL`. We found that our method is reasonably robust to these parameters. For the threshold θ in Definition 1, when we varied it from 0.1 to 0.35 on development set, the results remained the same and dropped a little if we used value larger than 0.35. We set it at 0.2 in our experiment. For the two parameters λ_1 and λ_2 in Equation 5, we set λ_1= 0.8 and λ_2 = 0.05 based on the results on the development set. Performance did not change much when we varied λ_1 from 0.5 to 1 and λ_2 from 0.05 to 0.1. We set λ = 0.2 in Equation 10; the performance nearly did not change when we varied it from 0.1 to 0.3.

The integration of classification based re-ranking method and graph-based method. The experiment is to study the two ways of integration in Section 4.3. Table 6 gives the results on `A-TUnion`. By comparing the results of `G(Cla)` with those of `Cla` in Tables 3, we found that graph-based method greatly improves the classification method `Cla` by using the result of `Cla` as the initial score of graph-based method. By comparing `Cla(G)` with `Cla` in Tables 3, we also found that using the results of graph-based methods as features can greatly improve method `Cla`. The reason for the improvement is that the integration can consider the relationship between candidate answers, while `Cla` alone does not consider the relationship between candidate answers.

Data	Method	All questions			Question with answers		
		P@1(#)	MRR	MAP	P@1	MRR	MAP
A-T	G(Cla)	0.652(1,000)	0.707	0.676	0.799	0.868	0.830
Union	Cla(G)	0.673(1,033)	0.725	0.691	0.826	0.890	0.848

Table 6: Integration of graph-based method and classification

Forum	All questions	Subset
TripAdvisor	2,788,701	1,031,245
LonelyPlanet	194,672	59,242
BootsnAll	74,105	19,589

Table 7: The number of question answer pairs extracted

The effectiveness of lexical mapping. The experiment is to evaluate the effect of lexical mapping between question and answer discussed in Section 4.3. The results are beyond our expectations: the learned lexical mapping did not help for all the three ranking methods (`CS`, `QL` and `KL`). Due to space limitation, the detailed results are ignored. After our analysis, we found that the lexical mapping is not always effective for forum data. For example, lexical mapping *how much* $\rightarrow$ *number* would be useful in TREC QA to locate answers. In our corpus, 31.2% correct answers for *how much* questions do not contain a number. One example of answer to *how much* questions is "*you can find it from the Website*." On the other hand, many answer candidates containing number are not real answers.

5.2.2 Results on more data

We applied our question detection method and answer detection method `G+KL` to the three forums that we crawled. The number of extracted question-answer pairs and its subset (the first question-answer pairs in each thread) is given in Table 7. We evaluated three methods on the three datasets. An annotator was asked to check the top-1 return results of the three methods. (To control workload, we did not request to check other returned answers, and thus we only report P@1 results.) The results are reported in Table 8. The number of all questions in each data is given below the name of data, and the number of questions in subsets in each data is 100. We observed the same trends for the three methods on the three data: both `KL` and `G+KL` outperform the baseline method `NA` and `G+KL` outperforms `KL` (statistically significant, p-value < 0.01). We also found that that the result on the subset is better than that on all questions as we observed on fully annotated data.

As a summary, our techniques are able to effectively extract question-answer pairs: 1) our question detection method outperformed rule-based methods and the method [18]; 2) our graph-based method outperformed a baseline, three IR methods and classification-based re-ranking; 3) the integration of graph-based model with classification method improved the classification-based re-ranking; 4) the lexical mapping did not help in our experiments.

6. CONCLUSIONS

In this paper, we described a new approach to extracting question-answer pairs from online forums. The extracted question-answer pairs could be used to enrich the knowledge base of community based QA service, instant answer service and Chatbot. Experimental results on real data show that our approach is effective. In the future, we will investigate the following problems: 1) to detect questions without answers; 2) to revisit more TREC QA techniques to see if they can help answer detection in forums [10]; 3) to model the relationship of questions in the same thread to improve answer detection, considering that each thread contain about 3 questions on average and they may share the same answer paragraph. In addition, we will also investigate the effectiveness of our techniques in passage retrieval on TREC QA data.

[10]In our corpus there are 10% factoid questions well studied in TREC QA

Data	Method	P@1(#) on all Qs	P@1(#) on subset
`A-Trip2` (167)	NA	0.611(102)	0.730(73)
	KL	0.647(108)	0.740(74)
	G+KL	**0.743**(124)	**0.860**(86)
`A-Lonely` (270)	NA	0.474(128)	0.666(68)
	KL	0.541(146)	0.745(76)
	G+KL	**0.622**(168)	**0.843**(86)
`A-Boots` (358)	NA	0.472(169)	0.730(73)
	KL	0.494(177)	0.740(74)
	G+KL	**0.612**(219)	**0.880**(88)

Table 8: The evaluation on other data

7. REFERENCES

[1] A. L. Berger, R. Caruana, D. Cohn, D. Freitag, and V. O. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proc. of SIGIR*, 2000.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of WWW7*, 1998.

[3] H. T. Dang, J. Lin, and D. Kelly. Overview of the trec 2007 question answering track. In *Proc. of TREC*, 2007.

[4] S. Ding, G. Cong, C.-Y. Lin, and X. Zhu. Using conditional random fields to extract contexts and answers of questions from online forums. In *ACL*, 2008.

[5] D. Feng, E. Shaw, J. Kim, and E. Hovy. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proc. of Intelligent user interfaces*, 2006.

[6] S. M. Harabagiu and A. Hickl. Methods for using textual entailment in open-domain question answering. In *Proc. of ACL*, 2006.

[7] J. Huang, M. Zhou, and D. Yang. Extracting chatbot knowledge from online discussion forums. In *Proc. of IJCAI*, 2007.

[8] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM*, 2005.

[9] V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In *CIKM*, 2005.

[10] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *SIGIR*, 2006.

[11] J. Ko, E. Nyberg, and L. Si. A probabilistic graphical model for joint answer ranking in question answering. In *SIGIR*, 2007.

[12] O. Kurland and L. Lee. Pagerank without hyperlinks: structural re-ranking using links induced by language models. In *SIGIR*, 2005.

[13] J. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, 2001.

[14] W. G. Lehnert. *The process of question answering: A computer simulation of cognition*. Lawrence Erlbaum Associates, 1978.

[15] J. Otterbacher, G. Erkan, and D. R. Radev. Using random walks for question-focused sentence retrieval. In *HLT/EMNLP*, 2005.

[16] J. Pei, J. Han, B. Mortazavi-Asl, and H. Pinto. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proc. ICDE*, 2001.

[17] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *ACL*, 2007.

[18] L. Shrestha and K. McKeown. Detection of question-answer pairs in email conversations. In *Proc. of COLING*, 2004.

[19] G. Sun, G. Cong, X. Liu, C.-Y. Lin, and M. Zhou. Mining sequential patterns and tree patterns to detect erroneous sentences. In *AAAI*, 2007.

[20] E. M. Voorhees and D. M. Tice. The trec-8 question answering track evaluation. In *TREC*, 1999.

[21] K. Zechner. Automatic generation of concise summaries of spoken dialogues in unrestricted domains. In *SIGIR*, 2001.

[22] D. Zhang and W. S. Lee. A language modeling approach to passage question answering. In *Proc. of TREC*, 2003.

[23] L. Zhou and E. H. Hovy. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proc. of ACL*, 2005.