Segmentation of Multi-Sentence Questions: Towards Effective Question Retrieval in cQA Services

Kai Wang, Zhao-Yan Ming, Xia Hu, Tat-Seng Chua
Department of Computer Science
School of Computing
National University of Singapore
{kwang, mingzy, huxia, chuats}@comp.nus.edu.sg

ABSTRACT

Existing question retrieval models work relatively well in finding similar questions in community-based question answering (cQA) services. However, they are designed for single-sentence queries or bag-of-word representations, and are not sufficient to handle multi-sentence questions complemented with various contexts. Segmenting questions into parts that are topically related could assist the retrieval system to not only better understand the user's different information needs but also fetch the most appropriate fragments of questions and answers in cQA archive that are relevant to user's query. In this paper, we propose a graph based approach to segmenting multi-sentence questions. The results from user studies show that our segmentation model outperforms traditional systems in question segmentation by over 30% in user's satisfaction. We incorporate the segmentation model into existing cQA question retrieval framework for more targeted question matching, and the empirical evaluation results demonstrate that the segmentation boosts the question retrieval performance by up to 12.93% in Mean Average Precision and 11.72% in Top One Precision. Our model comes with a comprehensive question detector equipped with both lexical and syntactic features.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Retrieval Models*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text Analysis*

General Terms

Algorithms, Design, Experimentation

Keywords

Question Answering, Question Segmentation, Question Matching, Yahoo! Answers

1. INTRODUCTION

Community-based Question Answering (cQA) services begin to emerge with the blooming of Web 2.0. They bring together a network of self-declared "experts" to answer questions posted by other people. Examples of these services include Yahoo! Answers (answers.yahoo.com) and Baidu Zhidao (zhidao.baidu.com) etc. Over times, a tremendous amount of historical QA pairs have been built up in their databases, and this transformation gives information seekers a great alternative to web search [2,18,19]. Instead of looking through a list of potentially relevant documents from the Web, users may directly search for relevant historical questions from cQA archives. As a result, the corresponding best answer could be explicitly extracted and returned. In view of the above, traditional information retrieval tasks like TREC [1] QA are transformed to similar question matching tasks [18,19].

There has been a host of work on question retrieval. The state-of-the-art retrieval systems employ different models to perform the search, including vector space model [5], language model [5,7], Okapi model [7], translation model [7,14,19] and the recently proposed syntactic tree matching model [18]. Although the experimental studies in these works show that the proposed models are capable of improving question retrieval performance, they are not well designed to handle questions in the form of multiple sub-questions complemented with sentences elaborating the context of the sub-questions. This limitation could be further viewed from two aspects. From the viewpoint of user query, the input to most existing models is simply a bag of keywords [5,19] or a single-sentence question [18]. It leads to a bottleneck in understanding the user's different information needs when the user query is represented in a complex form with many sub-questions. From the viewpoint of the archived questions, none of the existing work attempts to distinguish context sentences from question sentences, or tries to segment the archived question thread into parts that are topically based. It prevents the system from presenting the user the most appropriate fragments that are relevant to his/her queries.

Figure 1 illustrates an example of a question thread extracted from Yahoo! Answers. There are three sub-questions (Q_1, Q_2 and Q_3) asked in this thread, all in different aspects. If a user posts such example as a query, it is hard for existing retrieval systems to find all matches for the three sub-questions if the query is not well segmented. On the other hand, if a new similar query such as "*what are the requirements of being a dentist?*" is posted, it is also difficult for existing retrieval systems to return Q_3 as a valid match if Q_3 is not explicitly separated from its surrounding sub-questions and contexts. Given all these constraints, it is thus highly valuable and desirable to topically segment multi-sentence questions, and to properly align individual sub-questions with their context sentences. Good segmentation not only helps the question retrieval system to better analyze the user's complex information needs, but also assists it in matching the query with the most appropriate portions of the questions in the cQA archive.

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

C_1:	i heard somewhere that in order to become a dentist, you need certain hours of volunteering or shadowing.
Q_1:	**is that true?**
Q_2:	**if it is, how many hours?**
C_2:	i have only a few hours of such activity…
Q_3:	**and can you write down other requirements that one would need to become a dentist**
C_3:	i know there are a lot of things but if you can write down as much as you can, that'd be a lot of help.
C_4:	thanks

Figure 1: Example of multi-sentence questions extracted from Yahoo! Answers

It appears to be natural to exploit traditional text-based segmentation techniques to segment multi-sentence questions. Existing approaches to text segment boundary detection include similarity based method [3], graph based method [13], lexical chain based method [10], text tiling algorithm [6] and the topic change detection method [12] etc. Although experimental results of these segmentation techniques are shown to be encouraging, they mainly focus on general text relations and are incapable of modeling the relationships between questions and contexts. A question thread from cQA usually comes with multiple sub-questions and contexts, and it is desirable for one sub-question to be isolated from other sub-questions while closely linked to its context sentences.

After extensive study of the characteristics of questions in cQA archive, we introduce in this paper a new graph based approach to segment multi-sentence questions. The basic idea is outlined as follows. We first attempt to detect question sentences using a classifier built from both lexical and syntactic features, and use similarity and co-reference chain based methods to measure the closeness score between the question and context sentences. We model their relationships to form a graph, and use the graph to propagate the closeness scores. The closeness scores are finally utilized to group topically related question and context sentences.

The contributions of this paper are threefold: First, we build a question detector on top of both lexical and syntactic features. Second, we propose an unsupervised graph based approach for multi-sentence segmentation. Finally, we introduce a novel retrieval framework incorporating question segmentation for better question retrieval in cQA archives.

The rest of the paper is organized as follows: Section 2 presents the proposed technique for question sentence detection. Section 3 describes the detailed algorithm and architecture for multi-sentence segmentation, together with the new segmentation aided retrieval framework. Section 4 presents our experimental results. Section 5 reviews some related works and Section 6 concludes this paper with directions for future work.

2. QESTION SENTENCE DETECTION

Human generated content on the Web are usually informal, and it is not uncommon that standard features such as question mark or utterance are absent in cQA questions. For example, question mark might be used in cases other than questions (*e.g.* denoting uncertainty), or could be overlooked after a question. Therefore, traditional methods using certain heuristics or hand-crafted rules become inadequate to cope with various online question forms. To overcome these obstacles, we propose an automated approach to extracting salient sequential and syntactic patterns from question sentences, and use these patterns as features to build a question detector. Research on sequential patterns has been well discussed in many literatures, including the identification of comparative sentences [9], the detection of erroneous sentences [17] and question sentences [4]. However, works on syntactic patterns have only been partially explored [17,18]. Grounded on these previous works, we next explain our pattern mining process, together with the learning algorithm for the classification model.

2.1 Sequential Pattern Mining

Sequential Pattern is also referred to as *Labeled Sequential Pattern* (*LSP*) in the literatures. It is in the form of $S \Rightarrow C$, where S is a sequence $<t_1,...,t_n>$, and C is the class label that the sequence S is classified to. In the problem of question detection, a sequence is defined to be a series of tokens from sentences, and the class is in the binary form of {*Q, NQ*} (*resp*. question and non-question). The purpose of sequential pattern mining is to extract a set of frequent subsequence of words that are indicative of questions. For example, the word sequence "*anyone know what ... to*" is a good indication to characterize the question sentence "*anyone know what I can do to make me less tired*". Note that the mined sequential tokens need not to be contiguous as appeared in the original text.

There is a handful of algorithms available to find all frequent subsequences, and the *Prefixspan* algorithm [11] is reported to be efficient in discovering all relative frequents by using a pattern growth method. We adopt this algorithm in our work by imposing the following additional constraints:

1) Maximum Pattern Length: We limit the maximum number of tokens in a mined sequence to 5.
2) Maximum Token Distance: The two adjacent tokens t_n and t_{n+1} in the pattern need to be within a threshold window in the original text. We set it to 6.
3) Minimum Support: We set the minimum percentage of sentences in database D containing the pattern p to 0.45%.
4) Minimum Confidence: We set the probability of the pattern p being true in database D to 70%.

To overcome the sparseness problem, we generalize the tokens by applying Part-of-Speech (POS) taggers to all tokens except some keywords including 5W1H words, modal words, stop words and the most frequent occurring words mind from cQA such as "*any1*", "*im*", "*whats*" etc. For example, the pattern *<any1, know, what>* will be converted to *<any1, VB, what>*. Each generalized pattern makes up a binary feature for the classification model as we will introduce in Section 2.3.

2.2 Syntactic Shallow Pattern Mining

We found that sequential patterns at the lexical level might not always be adequate to categorize questions. For example, the lexical pattern *<when, do>* presumes the non-question "*Levator scapulae is used when you do the traps workout*" to be a question, and the question "*know someone with an eating disorder?*" could be missed out due to the lack of indicative lexical patterns. These limitations, however, could be alleviated by syntactic features. The tree pattern *(SBAR(WHADVP(WRB))(S(NP)(VP)))* extracted from the former example has the order of *NP* and *VP* being switched, which might indicate the sentence to be a non-question, whereas the tree pattern *(VP(VB)(NP(NP)(PP)))* may be evidence that the latter example is indeed a question, because this pattern is commonly observed in the archived questions.

Syntactic patterns have been partially explored in erroneous sentence detection [17], in which all non-leaf nodes are flattened for frequent substructure extraction. The number of patterns to be explored, however, grows exponentially with the size of the tree, which we think is inefficient. The reason is that the syntactic

pattern will become too specific if mining is extended to a very deep level, and nodes at certain levels do not carry much useful structural information favored by question detection (*e.g.*, the production rules *NP→DT•NN* at the bottom level).

For better efficiency, we focus only on certain portion of the parsing tree by limiting the depth of the sub-tree patterns to be within certain levels (*e.g.* $2 \le D \le 4$). We further generalize each syntactic pattern by removing some nodes denoting modifiers, preposition phrases and conjunctions etc. For instance, the pattern *SQ(MD)(NP(NN))(ADVP(RB))(VP(VP)(NP)(NP))* extracted from the question "*can someone also give me any advice?*" could be generalized into *SQ(MD)(NP(NN))(VP(VP)(NP)(NP))*, where the redundant branch *ADVP(RB)* that represents the adverb "*also*" is pruned. The pattern extraction process is outlined in Algorithm 1. The overall pattern mining strategy is analogous to the mining of sequential patterns, where the measures including *support* and *confidence* are taken into consideration to control the significance of the mined patterns. The discovered patterns are used together with the sequential patterns as features for the learning of classification model.

Algorithm 1 *ExtractPattern (S , D)*

Input: A set of syntactic trees for sentences (*S*); the depth range (*D*)
Output: A set of sub-tree shallow patterns extracted from *S*

```
1:  Patterns = {};
2:  for all Syntactic tree T ∈ S do
3:     Nodes ← level order traversal of T from top to bottom;
4:     for all node n ∈ Nodes do
5:        Extract subtree p rooted under node n, with depth within the range D;
6:        p ← generalize (p);        // remove modifier nodes etc.
7:        Patterns.add (p);          // add p as a candidate
8:     end for
9:  end for
10: return Patterns;
```

2.3 Model Learning

The input to an algorithm that learns a binary classifier consists normally of both positive and negative examples. While it is easy to discover certain patterns from questions, it becomes unnatural to identify characteristics for non-questions. The imbalanced data distribution leads normal classifiers to perform poorly on the model learning. To address this issue, we propose to learn with the one-class SVM method. One-class SVM is built on top of the standard two-class SVM method, and its basic idea is to transform features from only positive examples via a kernel to a hyper-plane, and treats the origin as the only member of the negative class. It further uses relaxation parameters to separate the image of positive class from the origin, and finally applies the standard two-class SVM techniques to learn a decision boundary. As a result, data points outside the boundary are considered to be outliers, i.e. non-questions in our problem.

The training data as used by traditional supervised learning methods usually require human labelling, which is not cheap. To save human efforts on data annotation, we take a shortcut by assuming all questions ending with question marks as an initial set of positive examples. This assumption is acceptable, as according to the results reported in [4], the rule-based method using only question mark achieves a very high precision (97%) in detecting questions. It in turn indicates that questions ending with "?" are highly likely to be real questions. To reduce the effect of possible outliers (*e.g.* non-questions ending with "?"), we need to purify the initial training set. There are many techniques available for training data refinement, such as bootstrapping, condensing, and editing. We choose a SVM-based data editing and classification method proposed by [15] to iteratively remove the samples likely to be outliers. The detail is not covered here as it is beyond the scope of this paper.

For one-class SVM training, the linear kernel is used, as it is shown to outperform other kernel functions. In the iterations of training data refinement, the parameter v that controls the upper bound percentage of outliers is set to 0.02. The question detector model learned ultimately serves as a component for the multi-sentence question segmentation system.

3. Multi-Sentence Question Segmentation

Unlike traditional text segmentation, question segmentation ought to group each sub-question with its context sentences while separating it from the other sub-questions. Investigations show that the user posting styles in the online environment are largely unpredictable. While some users ask multiple questions in an interleaved manner, some prefer to list the whole description first and ask all sub-questions later. Therefore, naive methods such as using distance based metrics will be inadequate, and it is a great challenge to segment multi-sentence questions especially when the description sentences in various aspects are mixed together.

In the remainder of this section, we present a novel graph-based propagation method for segmenting multi-sentence questions. While the graph based method has been successfully applied in many applications like web search, to the best of our knowledge, this is the first attempt to apply it to the question segmentation problem. The intuition behind the use of graph propagation approach is that if two description sentences are closely related and one is the context of a question sentence, then the other is also likely to be its context. Likewise, if two question sentences are very close, then the context of one is also likely to be the context of the other. We next introduce the graph model of the multi-sentence question, followed by the sentence closeness score computation and the graph propagation mechanism.

3.1 Building Graphs for Question Threads

Given a question thread comprising multiple sentences, we represent each of its sentences as a vertex v. The question detector is then applied to divide sentences into question sentences and non-question sentences (contexts), forming a question sentence vertex set V_q and a context sentence vertex set V_c respectively.

We model the question thread into a weighted graph (V, E) with a set of weight functions $w: E \Rightarrow \Re$, where V is the set of vertices $V_q \cup V_c$, E is the union of three edge sets $E_q \cup E_c \cup E_r$, and $w(E)$ is the weight associated with the edge E. The three edge sets E_q, E_c and E_r are respectively defined as follows:

- E_q: a set of *directed* edges $u \rightarrow v$, where $u, v \in V_q$;
- E_c : a set of *directed* edges $u \rightarrow v$, where $u, v \in V_c$;
- E_r : a set of *undirected* edges u–v, where $u \in V_q$ and $v \in V_c$.

While the undirected edge indicates the symmetric closeness relationship between a question sentence and a context sentence, the directed edge captures the asymmetric relation between two question sentences or two context sentences. The intuition of introducing the asymmetry relationship could be explained with the example given in Figure 1. It is noticed that C_1 is the context of the question sentence Q_1 and C_2 is the context of the question sentence Q_2. Furthermore, Q_2 is shown up to be motivated by Q_1, but not in the opposite direction. This observation gives us the sense that C_1 could also be the context of Q_2, but not for C_2 and Q_1. We may reflect this asymmetric relationship using the graph model by assigning higher weight to the directed edge $Q_1 \rightarrow Q_2$ than to $Q_2 \rightarrow Q_1$. As a result, the weight of the chain $C_1 \rightarrow Q_1 \rightarrow Q_2$ becomes much stronger than that of $C_2 \rightarrow Q_2 \rightarrow Q_1$, indicating that

C_1 is related to Q_2 but C_2 is not related to Q_1, which is consistent to our intuition. From another point of view, the asymmetry helps to regulate the direction of the closeness score propagation.

We give two different weight functions for edges depending on whether they are directed or not. For the directed edge ($u \rightarrow v$) in E_q and E_c, we consider the following factors in computing weight:

1) KL-divergence: given two vertices u and v, we construct the unigram language models M_u and M_v for the sentences they represent, and use KL-divergence to measure the difference between the probability distributions of M_u and M_v. We use $D_{KL}(M_u||M_v)$ to model the connectivity from u to v:

$$D_{KL}(M_u \,||\, M_v) = \sum_w p(w \,|\, M_u) \log \frac{p(w \,|\, M_u)}{p(w \,|\, M_v)} \quad (1)$$

Generally, the smaller the divergence value, the stronger the connectivity, and the value of $D_{KL}(M_u||M_v)$ is usually unequal to $D_{KL}(M_v||M_u)$, thereby representing the asymmetry.

2) Coherence: it is observed that the subsequence sentences are usually motivated by the earlier sentences. Given two vertices u, v, we say that v is motivated by u (or u motivates v) if v comes after u in the original post, and there are conjunction or linking words connecting in-between. The coherence score from u to v is determined as follows:

$$Coh(v \,|\, u) = \begin{cases} 1 & \text{if } v \text{ is motivated by } u \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

3) Coreference: coreference commonly occurs when multiple expressions in a sentence or multiple sentences have the same referent. We observe that sentences having the same referent are somehow connected, and the more the referents two sentences share, the stronger the connection. We perform the coreference resolution on a question thread, and measure the coreference score from vertex u to vertex v as follows:

$$Ref(v \,|\, u) = \begin{cases} 1 - e^{-|referent \in \{u \cap v\}|}, & \text{if } v \text{ comes after } u \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Note that all the metrics introduced above are asymmetric, meaning that the measure from u to v is not necessarily the same as that from v to u. Given two vertices u, $v \in E_q$ or E_c, the weight of the edge $u \rightarrow v$ is computed by a linear interpolation of the three factors as follows:

$$w_1(u \rightarrow v) = \alpha_1 \frac{1}{1 + D_{KL}(M_u \,||\, M_v)} + \alpha_2 Coh(v \,|\, u) + \alpha_3 Ref(v \,|\, u)$$

where $0 \le \alpha_1, \alpha_2, \alpha_3 \le 1$. (4)

Since $D_{KL}(M_v||M_u) \ge 0$, $0 \le Coh(v|u) \le 1$, and $0 \le Ref(v|u) \le 1$, the interval range of $w_1(u \rightarrow v)$ is between 0 to 1, and we do not need to apply normalization on this weight. We employed grid search with 0.05 stepping space in our experiments and found that the combination of $\{\alpha_1 = 0.4, \alpha_2 = 0.25, \alpha_3 = 0.35\}$ gives the most satisfactory results.

While the weight of the directed edges in E_q and E_c measures the throughput of the score propagation from one to another, the weight of the undirected edge (u–v) in E_r demonstrates the true closeness between a question and a context sentence. We consider the following factors in computing the weight for edges in E_r:

1) Cosine Similarity: given a question vertex u and a context vertex v, we measure their cosine similarity weighted by the word inverse document frequency (idf_w) as follows:

$$Sim(u,v) = \frac{\sum_{w \in u,v} f_u(w) \times f_v(w) \times (idf_w)^2}{\sqrt{\sum_{w \in u} (f_u(w) \times idf_w)^2} \sqrt{\sum_{w \in v} (f_v(w) \times idf_w)^2}} \quad (5)$$

where $f_u(w)$ is the frequency of word w in sentence u, idf_w is the inverse document frequency (# of posts containing w). We do not employ KL-divergence as we believe that the similarity between question and context sentences is symmetric.

2) Distance: questions and contexts separated far away are less likely to be relevant as compared to neighboring pairs. Hence, we take the following distance factor into account:

$$Dis(u,v) = e^{-\delta(u,v)} \quad (6)$$

where $\delta(u,v)$ is proportional to the number of sentences between u and v in the original post.

3) Coherence: the coherence between a question and a context sentence is also important, and we take it into account with the exception that the order of appearance is not considered:

$$Coh(u,v) = \begin{cases} 1 & \text{if linked by conjunction words} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

4) Coreference: similarly, it measures the number of the same referents in the question and context, without considering their ordering:

$$Ref(u,v) = 1 - e^{-|referent \in \{u \cap v\}|} \quad (8)$$

The final weight of the undirected edge (u–v) is computed by a linear interpolation of the abovementioned factors:

$$w_2(u - v) = \beta_1 Sim(u,v) + \beta_2 Dis(u,v) + \beta_3 Coh(u,v) + \beta_4 Ref(u,v)$$

where $0 \le \beta_1, \beta_2, \beta_3, \beta_4 \le 1$ (9)

The combination of $\{\beta_1 = 0.4, \beta_2 = 0.1, \beta_3 = 0.3, \beta_3 = 0.2\}$ produces best results with grid search. Note that normalization is not required as each factor is valued between 0 and 1. With the weight of each edge defined, we next introduce the propagation mechanism of the edge scores.

3.2 Propagating the Closeness Scores

For each pair of vertices, we assign the initial closeness score to be the weight of the edge in-between using the weight function introduced in Section 3.1, depending on whether the edge is in E_q, E_c or E_r. Note that if the edge weight is very low, two sentences might not be closely related. For fast processing, we use a weight threshold θ to prune edges with weight below θ. The parameter θ is empirically determined, and we found in our experiments that the results are not very sensitive to θ value below 0.15.

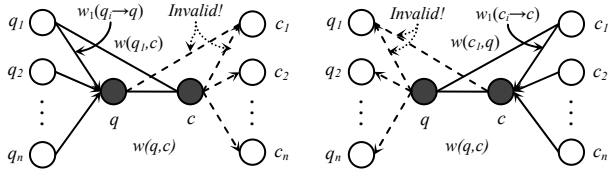
Algorithm 2 *MapPropagation (G(V,E))*

Input: The map model with initial scores assigned to every edge
Output: The map with updated closeness scores between questions and contexts

```
for every context c∈V_c and every question q∈V_q do  // initialization
    w(q,c) = w_2(q,c);
end for
while score is not converged do
    for every context c∈V_c and question q∈V_q do  // propagate from c to q
        w'(q,c) = MAX_{q_i ∈ V_q} { λw(q_i,c)w_1(q_i→q) }
        if (w(q,c) < w'(q,c))
            w(q,c) = w'(q,c)
    end for
    for every question q∈V_q and context c∈V_c do  // propagate from q to c
        w'(c,q) = MAX_{c_i ∈ V_c} { λw(c_i,q)w_1(c_i→c) }
        if (w(c,q) < w'(c,q))
            w(c,q) = w'(c,q)
    end for
end while
```

With the initial closeness scores, we carry out the score propagation using the algorithm outlined in Algorithm 2. The basic idea of this propagation algorithm is that, given a question sentence q and a context sentence c, if there is an intermediate question sentence q_i such that the edge weight $w_1(q_i \rightarrow q)$, together with the closeness score $w(q_i,c)$ between q_i and c, are both

relatively high, then the closeness score $w(q,c)$ between q and c could be updated to $\lambda w_1(q_i \rightarrow q) w(q_i,c)$ in case the original score is lower than that. In other words, q_i becomes the evidence that q and c are related. The propagation algorithm works similarly in propagating scores from question sentences to context sentences, where an intermediate context c_i could be the evidence that c and q are related. Notice that the direction of propagation is not arbitrary. For example, it makes no sense if we propagate the score along the path of $c \rightarrow c_i \rightarrow q$, because c_i is simply the receiver of c, which could not be the evidence that a question and a context are correlated. When considering a pair of q and c, the possible directions of propagation are illustrated in Figure 2, in which the dashed lines indicate invalid propagation paths.



Propagating from c to q
only nodes in V_q are considered to be intermediate nodes for propagation

Propagating from q to c
only nodes in V_c are considered to be intermediate nodes for propagation

Figure 2: Illustration of the direction of score propagation

The damping factor λ in the algorithm controls the transitivity among nodes. In some circumstances, the propagated closeness score might not indicate the true relatedness between two nodes, especially when the score is propagated through an extremely long chain. For example, $\{ABC\}$ is close to $\{BCD\}$, $\{BCD\}$ is close to $\{CDE\}$, and $\{CDE\}$ is close to $\{DEF\}$. The propagation chain could infer $\{ABC\}$ to be related to $\{DEF\}$, which is not true. The introduction of damping factor λ can leverage this propagation issue by penalizing the closeness score when the chain becomes longer. We empirically set λ to 0.88 in this work.

The propagation of the closeness score will eventually converge. This is controlled by our propagation principle that the updated closeness score is a multiplication of two edge weights whose value is defined to fall between 0 and 1. Hence the score is always upper bounded by the maximum weight of the edges in E.

After the propagation reaches the stationary condition, we need to extract all salient edges in E_r for the alignment of questions and contexts. One straightforward method is to pre-define a threshold ψ, and remove all edges weighted under ψ. However, this method is not very adaptive, as the edge weights vary greatly for different questions and a pre-defined threshold is not capable to regulate the appropriate number of alignments between questions and contexts. In this work, we take a dynamical approach instead: we first sort edges in E_r by the closeness score and extract them one by one in descending order $<e_1, e_2, \ldots, e_n>$. The extraction process terminates at e_m when one of the following criteria is met:

1. $ew_m - ew_{m+1} > \omega(\frac{1}{m}\sum_{i=1}^{m} ew_i - ew_m)$, where ew_i is the i-th edge weight in the order and ω is the control parameter.
2. $ew_{m+1} < \eta$, where η is a pre-defined threshold controlling the overall connection quality (we set it to 0.05).
3. $m = n$, meaning all edges have been extracted out from E_r.

When the extraction procedure terminates, the extracted edge set $\{e_1,\ldots,e_m\}$ represents the final alignment between questions and contexts. For each edge e_i connecting between a context c and a question q, c will be considered as the context to question q, and they belong to the same question segment. For example, a final edge set $\{(q_1,c_1), (q_2,c_2), (q_1,c_2), (q_2,c_4), (q_3,c_1), (q_2,c_3)\}$ produces three question segments: $(q_1 - c_1,c_2)$, $(q_2 - c_2,c_3,c_4)$ and $(q_3 - c_1)$. Note that the segmentation works in a fuzzy way such that no explicit boundaries are defined between sentences. Instead, a question could have multiple context sentences, whereas a context sentence does not necessarily belong to only one question.

3.3 Segmentation-aided Retrieval

By applying segmentation on the multi-sentence questions from cQA, sub-questions and their corresponding contexts that are topically related could be grouped. Figure 3 shows an improved retrieval framework with segmentation integrated. Different from existing models, the question matcher matches two question sentences with the assistance of additional related contexts such that the users' query can be matched with the archived cQA questions more precisely. More specifically, the user query is no longer restricted to a short single-sentence question, but can be in the form of multiple sub-questions complemented with many description sentences. An archived question thread asking in various aspects could also be indexed into different question-context pairs such that the matching is performed on the basis of each question-context pair.

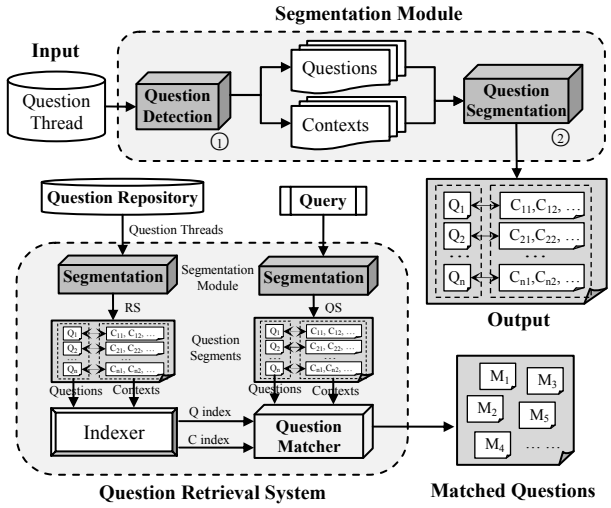


Figure 3: Retrieval framework with question segmentations

4. EXPERIMENTS

In this section, we present empirical evaluation results to assess the effectiveness of our question detection model and multi-sentence segmentation technique. In particular, we conduct experiments on the Yahoo! Answers QA archive and show that our question detection model outperforms traditional rule based or lexical based methods. We further show that our segmentation model works more effectively than conventional text segmentation techniques in segmenting multi-sentence questions, and it gives additional performance boosting to cQA question matching.

4.1 Evaluation of Question Detection

Dataset: We issued *getByCategory* API query to Yahoo! Answers, and collected a total of around 0.8 million question threads from *Healthcare* domain. From the collected data, we generate the following three datasets for the experiments:

- Pattern Mining Set: Around 350k sentences extracted from 60k question threads are used for lexical and syntactic pattern

mining, where those ending with "?" are treated as question sentences and the others as non-question sentences[1].

- Training Set: Around 130k sentences ending with "?" from another 60k question threads are used as the initial positive examples for one-class SVM learning method.
- Testing Set: Two annotators are asked to tag some randomly picked sentences from a third post set. A total of 2004 question sentences and 2039 non-question sentences are annotated.

Method: To evaluate the performance of our question detection model, we use five different systems for comparison:

1) 5W1H (baseline1): a rule based method determines that a sentence is a question if it contains 5W1H words.
2) Question Mark (baseline2): a rule based method judges that a sentence is a question if it ends with the question mark "?".
3) SeqPattern: Using only sequential patterns as features.
4) SynPattern: Using only syntactic patterns as features.
5) SeqPattern+SynPattern: Using both sequential patterns and syntactic patterns as features for question sentence detection.

A grid search algorithm is performed to find the optimal number of features used for model training, and a set of 1314 sequential patterns and 580 syntactic patterns are shown to give the best performance. Table 1 illustrates some pattern examples mined.

Table 1: Examples for sequential and syntactic patterns

Pattern Type	Pattern Example	
Sequential Pattern	*< anyone VB NN >*	*< what NN to VB NN >*
	< NNS should I >	*< can VB my NN>*
	< JJS NN to VB >	…
Syntactic Pattern	*(SBARQ (CC)(WHADVP (WRB))(SQ (VBP)(NP)(VP)))*	
	(SQ (VBZ)(NP (DT))(NP (DT)(JJ)(NN)))	
	(VP (VBG)(S (VP)))	…

Metrics & Results: We employ Precision, Recall, and F_1 as metrics to evaluate the question detection performance. Table 2 tabulates the comparison results. From the table, we observe that 5W1H performs poorly in both precision and recall. Question mark based method gives the highest precision, but the recall is relatively low. This observation is in line with the reported results in [4]. On the other hand, SeqPattern gives relatively high recall and SynPattern gives relatively high precision. The combination of both augments the performance in both precision and recall by a lot, and it achieves statistically significant improvement (t-test, p-value<0.05) as compared to SeqPattern and SynPattern. We believe that the improvement stems from the ability of the detection model to capture the salient characteristics in questions at both the lexical and syntactic levels. The results are also consistent with our intuition that sequential patterns could misclassify a non-question to a question, but syntactic patterns may leverage it to certain extent. It is noted that our question detector exhibits a sufficiently high F_1 score for its use in the multi-sentence question segmentation model in the later phase.

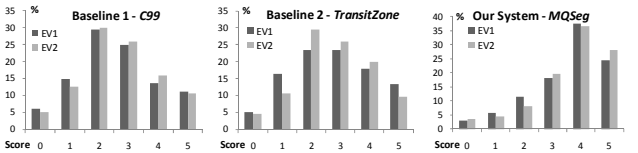
Table 2: Performance comparisons for question detection on different system combinations

System Combination	Precision (%)	Recall (%)	F_1(%)
(1) 5W1H	75.37	49.50	59.76
(2) Question Mark	**94.12**	77.50	85.00
(3) SeqPattern	88.92	88.47	88.69
(4) SynPattern	90.06	78.19	83.71
(5) SeqPattern+SynPattern	**92.11**	**89.67**	**90.87**

[1] This is acceptable for a large dataset, as a question ending with "?" is claimed to have high precision to be a true question.

4.2 Direct Assessment of Multi-Sentence Question Segmentation via User Study

We first evaluate the effectiveness of our multi-sentence question segmentation model (denoted as *MQSeg*) via a direct user study. We set up two baselines using the traditional text segmentation techniques for comparison. The first baseline (denoted as *C99*) employs the *C99* algorithm [4], which uses a similarity matrix to generate a local sentence classifier so as to isolate topical segments. The second baseline (denoted as *TransitZone*) is built on top of the method proposed in [12]. It measures the thematic distance between sentences to determine a series of transition zones, and uses them to locate the boundary sentences. To conduct the user study, we generate a small dataset by randomly sampling 200 question threads from the collected data. We run the three segmentation systems for each question thread, and present the segmentation results to two evaluators without telling them from which system the result was generated. The evaluators are then asked to rate the segmentation results using a score from 0 to 5 with respect to their satisfaction. Figure 4 shows the score distributions from the evaluators for three different segmentation systems. We can see from Figure 4 that users give relatively moderate scores (avg. 2 to 3) to the results returned by two baseline systems, whereas they seem to be more satisfied with the results given by *MQSeg*. The score distribution in *MQSeg* largely shifts towards high end as compared to the two baseline systems. The average rating scores for three different systems are 2.63, 2.74, and 3.6 respectively. We consider two evaluators to be agreeable to the segmentation result if their score difference does not exceed 1, and the average level of peer agreement obtained between the two evaluators is 93.5%.



Avg	StDev	Peer Agrmt
2.63	1.34	93.5%

Avg	StDev	Peer Agrmt
2.74	1.35	92.5%

Avg	StDev	Peer Agrmt
3.6	1.28	94.5%

Figure 4: Score distribution of user evaluation for 3 systems

It is to our expectation that *MQSeg* performs better than *C99* or *TransitZone* segmentation systems. One straightforward reason is that *MQSeg* is specifically designed to segment multi-sentence questions, whereas the traditional systems are designed for generic purpose and do not distinguish question sentences from contexts. While the conventional systems fail to capture the relationship between questions and their contexts, our system aligns the questions and contexts in a fuzzy way that one context sentence could belong to different question segments. As online content is usually freely posted and does not strictly adhere to the formal format, we believe that our fuzzy grouping mechanism is more suitable to correlate sub-questions with their contexts, especially when there is no obvious sign of association.

4.3 Performance Evaluation on Question Retrieval with Segmentation Model

In cQA, either archived questions or user queries could be in the form of a mixture of question and description sentences. To further evaluate our segmentation model and to show that it can improve question retrieval, we set up question retrieval systems coupled with segmentation modules for either question repository or user query.

Methods: We select BoW, a simple bag-of-word retrieval system that matches stemmed words between the query and questions, and STM, a syntactic tree matching retrieval model proposed in [18] as two baseline systems for question retrieval. For each baseline, we further set up three different combinations:

1) Baseline+RS: a baseline retrieval system integrated with question repository segmentation.
2) Baseline+QS: a baseline retrieval system equipped with user query segmentation.
3) Baseline+RS+QS: the retrieval system with segmentations for both repository questions and user queries.

It gives rise to a total of 6 different combinations of methods for comparison.

Dataset: We divide the collected 0.8 million question dataset from Yahoo! Answers into two parts. The first part (0.75M) is used as a question repository, while the remaining part (0.05M) is used as a test set. For data preprocessing, systems coupled with RS will segment and index each question thread in the repository accordingly, whereas systems without RS simply performs basic sentence indexing tasks. From the test set, we randomly select 250 sample questions, each of which is in the form of one single-sentence question with some context sentences. The reason that we do not take queries of multi sub-questions as test cases is that traditional cQA question retrieval systems cannot handle complex queries, making it impossible to conduct the comparison test. Nevertheless, it is sufficient to use single-question queries here as our purpose is to testify that the context extracted by the segmentation model could help question matching.

For systems equipped with user query segmentation (QS), we simply use the testing samples as they are, whereas for systems without QS, we manually extract the question sentences from the samples and use them as queries without their corresponding context sentences. For each retrieval system, the top 10 retrieval results are kept. For each query, we combine the retrieval results from different systems, and ask two annotators to label each result to be either "relevant" or "irrelevant" without telling them from which system the result is generated. The kappa statistic for identifying relevance between two evaluators is reported to be 0.73. A third person will be involved if conflicts happen. By eliminating some queries that have no relevant matches, the final testing set contains 214 query questions.

Table 3: Performance of different systems measured by MAP, MRR, and P@1 (%chg shows the improvement as compared to BoW or STM baselines. All measures achieve statistically significant improvement with t-test, p-value<0.05)

Systems	MAP	%chg	MRR	%chg	P@1	%chg
BoW	0.5807	–	0.7138	–	0.5981	–
BoW+RS	0.6389	10.02	0.7565	5.98	0.6542	9.38
BoW+QS	0.6245	7.54	0.7429	4.07	0.6355	6.25
BoW+RS+QS	0.6558	12.93	0.7690	7.73	0.6682	11.72
STM	0.6653	–	0.7429	–	0.6308	–
STM+RS	0.7310	9.88	0.7774	4.64	0.6776	7.41
STM+QS	0.7238	8.79	0.7893	6.24	0.6916	9.63
STM+RS+QS	0.7415	11.45	0.7984	7.46	0.7009	11.11

Metrics & Results: We evaluate the performance of retrieval systems using three metrics: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Precision at Top One (P@1). The evaluation results are presented in Table 3.

We can see from Table 3 that STM consistently outperforms BoW. Applying question repository segmentation (RS) over both BoW and STM baselines boosts system performance by a lot. All RS coupled systems achieve statistically significant improvement in terms of MAP, MRR and P@1. We believe that the improvement stems from the ability of the segmentation module to eliminate irrelevant content that is favored by traditional BoW or STM approaches. Take the query question "*What can I eat to put on weight?*" as an example, traditional approaches may match it to an irrelevant question "*I'm wearing braces now. what am I allowed to eat?*" due to their high similarity on the questioning part. The mismatch however, could be alleviated if repository segmentation gets involved, where the context sentence can give clear clue that the above archived sentence is not relevant to the user query.

Performing user query segmentation (QS) on top of baseline systems also brings in large improvements in all metrics. This result is in line with our expectation. The introduction of QS is based on the intuition that contexts could complement questions with additional information, which help the retrieval system to better understand the user's information need. For example, given an example question from our testing set "*Questions about root canal?*", it makes no sense for retrieval systems to find its related questions if the context is absent, because there could be hundreds of irrelevant questions in the QA archive as long as they are concerned about "*root canal*".

Interestingly, STM+QS gives more improvement over STM as compared to BoW+QS over BoW. Our reading is that, BoW is less sensitive to the query context as compared to STM. To be more specific, the query context provides information at the lexical level, and BoW handles bad-of-word queries at the lexical level, whereas STM matches questions at the syntactic level. As such, it is reasonable that matching at both lexical and syntactic levels (STM+QS) gives more performance boosting as compared to only at lexical level (BoW+QS). Similar interpretation could be applied to explain the finding that BoW+RS system gives more significant improvement over BoW as compared to BoW+QS. Furthermore, we conjecture that, without RS, BoW is likely to match the query with some context sentences, whereas having question repository properly segmented overcomes this issue to a large extent.

Lastly, the combination of both RS and QS brings in significant improvement over the other methods in all metrics. The MAP on systems integrated with RS and QS improves by 12.93% and 11.45% respectively over BoW and STM baselines. RS+QS embedded systems also yield better top one precision by correctly retrieving questions at the first position on 143 and 150 questions respectively, out of a total of 214 questions. These significant improvements are consistent to our observations that RS and QS complement each other in not only better analyzing the user's information need but also organizing the question repository more systematically for efficient question retrieval.

Error Analysis: Although we have shown that RS together with QS improves question retrieval, there is still plenty of room for improvement. We perform micro-level error analysis and found that the segmentation sometimes fails to boost retrieval performance mainly for the following three reasons:

1) Question detection error: The performance of question segmentation highly depends on the reliability of the question detector. Although we have shown that the performance of our question detection model is very competitive, the noisy online environment still leads many questions to be miss-detected. Examples are the question in abbreviated form such as "*signs of a concussion?*" and the question in declarative form such as "*I'm going through some serious insomniac issues?*" etc.

2) Closeness gaps: The true closeness score between sentences is relatively hard to measure. For simplicity and efficiency, the relatedness measure in this work is more at the lexical level, and the only semantic factor we have taken is coreference resolution. These measures may become insufficient when the sentences grow in complexity, especially when there is a lack of lexical evidence (*e.g.* cue words or phrases etc.) indicative of the connection between two sentences. This is a difficult challenge, and a good strategy may be to apply more advanced NLP techniques or semantic measures.
3) Propagation errors: The propagated closeness score could be unreliable even when the propagation chain is short. Given three questions "*is it expensive to see a dentist instead?*" (Q_1), "*if it is not, how long it takes to get my teeth whitened?*" (Q_2), and "*How many ways to get my teeth whitened?*" (Q_3), Q_1 is considered to be the predecessor of Q_2, and Q_3 is closed to Q_2, but the linkage between Q_1 and Q_3 is so weak that assigning the context of Q_1 to Q_3 becomes inappropriate. We conjecture that selecting the damping factor λ in a more dynamic way (*e.g.* associating λ with the actual question) could help to adjust the propagation trend. We leave it to our future work.

5. RELATED WORK

There have been many literature works in the direction of question retrieval, and these works could generally be classified into two genres: the early FAQ retrievals and the recent cQA retrievals. Among FAQ related works, many retrieval models have been proposed, including the conventional vector space model [8], noisy channel model [16], and translation based model [14] etc. Most of these works tried to extract a large number of FAQ pairs from the Web, and use the FAQs dataset to do training and retrieval.

The cQA archive is different from FAQ collections in the sense that the content of cQA archive is much noisier and the scope is much wider. The state-of-the-art cQA question retrieval systems also employ different models to perform the search, including the vector space model [5], language model [5,7], Okapi model [7], and translation model [7,14,19] etc. Claiming that purely lexical level models are not adequate to cope with natural languages, Wang et al. [18] proposed a syntactic tree matching model to rank historical questions.

However, all these previous works handle bag-of-words queries or single-sentence questions only. On the contrary, we take a new approach by introducing a question segmentation module, where the enhanced retrieval system is capable of segmenting a multi-sentence question into parts that are topically related and perform better question matching thereafter. To the best of our knowledge, no previous work has attempted to look into this direction, or use question segmentation to improve the question search.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a new segmentation approach for segmenting multi-sentence questions. It separates question sentences from non-question sentences and aligns them according to their closeness scores as derived from the graph based model. The user study showed that our system produces more satisfactory results as compared to the traditional text segmentation systems. Experiments conducted on the cQA question retrieval systems further demonstrated that segmentation significantly boosts the performance of question matching.

Our qualitative error analysis revealed that the segmentation model could be improved by incorporating a more robust question detector, together with more advanced semantic measures. One promising direction for future work would be to also analyze the answers to help question segmentation. This is because answers are usually inspired by questions, where certain answer patterns could be helpful to predict the linkage between question and context sentences. The segmentation system in this work takes all noisy contexts as they are, without further analysis. The model could be further improved by extracting the most significant content and align them with question sentences. Finally, it is important to evaluate the efficiency of our proposed approach as well as to conduct additional empirical studies of the performance of question search with segmentation model incorporated.

7. REFERENCES

[1] Trec proceedings. http://trec.nist.gov/.
[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM*, 2008.
[3] F. Y. Y. Choi. Advances in domain independent linear text segmentation. In *NAACL*, 2000.
[4] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun. Finding question-answer pairs from online forums. In *SIGIR*, 2008.
[5] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *HLT-ACL*, 2008.
[6] M. A. Hearst. Multi-paragraph segmentation of expository text. In *ACL*, 1994.
[7] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM*, 2005.
[8] V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In *CIKM*, 2005.
[9] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *SIGIR*, 2006.
[10] M.-Y. Kan, J. L. Klavans, and K. R. McKeown. Linear segmentation and segment significance. In *WVLC*, 1998.
[11] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, and M. chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.
[12] V. Prince and A. Labadi´e. Text segmentation based on document understanding for information retrieval. 2007.
[13] J. C. Reynar. Topic segmentation: Algorithms and applications, 1998.
[14] S. Riezler, A. Vasserman, I. Tsochantaridis, V. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *ACL*, 2007.
[15] X. Song, G. Fan, and M. Rao. Svm-based data editing for enhanced one-class classification of remotely sensed imagery. *Geoscience and Remote Sensing Letters, IEEE*, 2008
[16] R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In *HLT-NAACL*, 2004.
[17] 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.
[18] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*, 2009.
[19] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR*, 2008.