A Syntactic Tree Matching Approach to Finding Similar Questions in Community-based QA Services

Kai Wang
Department of Computer Science
School of Computing
National University of Singapore
kwang@comp.nus.edu.sg

Zhaoyan Ming
Department of Computer Science
School of Computing
National University of Singapore
mingzy@comp.nus.edu.sg

Tat-Seng Chua
Department of Computer Science
School of Computing
National University of Singapore
chuats@comp.nus.edu.sg

ABSTRACT

While traditional question answering (QA) systems tailored to the TREC QA task work relatively well for simple questions, they do not suffice to answer real world questions. The community-based QA systems offer this service well, as they contain large archives of such questions where manually crafted answers are directly available. However, finding similar questions in the QA archive is not trivial. In this paper, we propose a new retrieval framework based on syntactic tree structure to tackle the similar question matching problem. We build a ground-truth set from Yahoo! Answers, and experimental results show that our method outperforms traditional bag-of-word or tree kernel based methods by 8.3% in mean average precision. It further achieves up to 50% improvement by incorporating semantic features as well as matching of potential answers. Our model does not rely on training, and it is demonstrated to be robust against grammatical errors as well.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Experimentation

Keywords

Question Answering, Syntactic Structure, Question Matching, Yahoo! Answers

1. INTRODUCTION

Traditional TREC QA task has made significant progress since it was first introduced in 1990s [1]. However, research on TREC QA has largely targeted on short, factoid-based, questions, for which concise answers are expected. For example, TREC QA simply expects the year "*1960*" for the simple question "*In what year did Sir Edmund Hillary search for Yeti?*". It was earlier claimed that while QA systems tailored to the TREC QA task worked relatively well for factoid-type questions, they might not be necessarily effective in question answering applications outside TREC [7]. In real world, more complex questions are usually asked, and users are more willing to obtain a longer and more comprehensive answer which contains sufficient context information. Traditional QA systems are now facing problems of being deployed into real world.

With the blooming of Web 2.0, social collaborative applications such as Wikipedia, YouTube, Facebook etc. begin to flourish, and there have been an increasing number of Web information services that bring together a network of self-declared "experts" to answer questions posted by other people. This is referred to as the community-based question answering services (cQA). In these communities, anyone can ask and answer questions on any topic, and people seeking information are connected to those who know the answer. As answers are usually explicitly provided by human and are of high quality, they can be helpful in answering real world questions.

Yahoo! Answers, launched on December 13, 2005, is now becoming the largest knowledge-sharing online community among several popular cQA services. Over times, a tremendous number of previous QA pairs have been stored in its database, and in most circumstances, users may directly get the answers from Yahoo! Answers by searching from this QA archive, rather than looking through a list of potentially relevant documents from the Web. As such, instead of extracting answers from a certain document corpus, the retrieval task in cQA becomes the task of finding relevant similar questions with new queries.

The similar question matching task is, however, not trivial. One of the major reasons is that instead of inputting just keywords or so, users form questions using natural language, where questions are encoded with various lexical, syntactic and semantic features. For example, "*how can I lose weight in a few month?*" and "*are there any ways of losing pound in a short period?*" are two similar questions asking for methods of losing weight, but they neither share many common words nor follow identical syntactic structure. This gap makes the similar question matching task difficult. Similarity measure techniques based purely on the bag-of-word (BoW) approach may perform poorly and become ineffective in these circumstances.

Syntactic or semantic features hence become vital for such task. The tree kernel function [5] is one of the most effective ways to represent the syntactic structure of a sentence. In general, it divides the parsing tree into several sub-trees and computes the inner product between two vectors of sub-trees. Although there

SIGIR'09, July 19–23, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-483-6/09/07...$5.00.

have been some successful applications using it, like Question Classification [3,13,19], the tree kernel-like function has not been directly applied to finding similar questions in the QA archive. Moreover, its matching scheme is too strict to be directly employed to our question matching problem. In this paper, we re-formulate the tree kernel framework, and introduce a new retrieval model to find similar questions. We extensively study the structural representations of questions to encode not only lexical but also syntactic and semantic features into the matching model. Our model does not rely on training, and it is shown to be robust against grammatical errors as well.

The rest of the paper is organized as follows: Section 2 gives a background introduction on the well-known tree kernel concept. Section 3 presents the architecture of our syntactical tree matching model. Section 4 describes an improved model with semantic features incorporated. Section 5 presents our experimental results. Section 6 reviews some related works and Section 7 concludes our paper with directions for future works.

2. BACKGROUND ON TREE KERNEL

Traditional information retrieval tasks adopt the BoW or language model etc. to perform retrieval. However, these purely lexical based approaches are often inadequate to perform fine-level textual analysis if the task involves the use of more varying syntactic structures or complex semantic meanings.

In order to utilize more structural or syntactical information and capture higher order dependencies between grammar rules, Collins tried to consider all tree fragments that occur in a parsing tree [5]. He defined the tree fragment to be any sub-tree that includes more than one node, with the restriction that the entire rule productions must be included. Zhang & Lee [19] inherited it by proposing a slightly different definition, in which all terminal symbols are included into sub-trees, arguing that the tree kernel can back off to the word linear kernel.

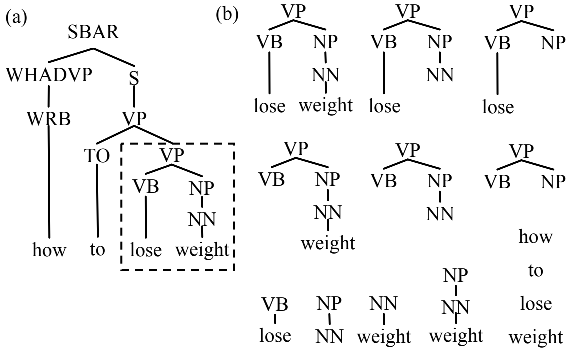


Figure 1. (a) The Syntactic Tree of the Question "How to lose weight?". (b) Tree Fragments of the Sub-tree covering "lose weight"

Figure 1 gives an illustration on how the tree decomposition works according to Zhang & Lee's definition. Figure 1(a) shows the entire syntactic parsing tree of the question "*How to lose weight?*", and Figure 1(b) shows all the sub-trees under the node of VP covering the phrase "*lose weight*". All the tree fragments produced contain the entire production rule, i.e., any sub-trees containing a part of the production rule such as "*VP→VB*" for "*VP→VB·NP*" are considered invalid.

The tree kernel was designed based on the idea of counting the number of tree fragments that are common to both parsing trees, and it could be defined as:

$$k(T_1,T_2)=\sum_{n_1\in N_1}\sum_{n_2\in N_2}C(n_1,n_2)\ , \qquad (1)$$

where N_1 and N_2 are sets of nodes in two syntactic trees T_1 and T_2, and $C(n_1,n_2)$ equals to the number of common fragments rooted in nodes n_1 and n_2. However, to enumerate all possible tree fragments is an intractable problem. The tree fragments are thus implicitly represented, and with dynamic programming, the value of $C(n_1,n_2)$ can be efficiently computed as follows:

$$C(n_1,n_2)=\begin{cases}0,\ \text{if } n_1\neq n_2\\ 1,\ \text{if } n_1=n_2 \text{ and they are terminal nodes}\\ \lambda,\text{if } n_1=n_2 \text{ and they are pre - terminal nodes}\\ \lambda\prod_{j=1}^{nc(n_1)}[1+C(ch(n_1,j),ch(n_2,j))],\ \text{otherwise}\end{cases} \qquad (2)$$

where $nc(n)$ is the total number of children of node n and $ch(n,j)$ is the *j-th* child of node n in the tree. $n_1{=}n_2$ denotes that the labels and production rules of node n_1 and n_2 are the same, and $n_1{\neq}n_2$ denotes the opposite. The parameter λ, a weighing factor, is used to resolve the kernel peaking problem.

3. SYNTACTIC TREE MATCHING

Although the tree kernel function has been successfully applied in some areas like question classification, there is no precedent work of using it to help find similar questions. The tree kernel metric measures the distance between two sentences, but there are two major limitations that prevent it from being employed directly in our question matching problem: (a) the tree kernel function merely relies on the intuition of counting the common number of sub-trees, whereas the number might not be a good indicator of the similarity between two questions; and (b) the two evaluated sub-trees have to be identical to allow further parent matching, for which semantic representations cannot fit in well. To remedy the second issue, the *Shallow Semantic Tree Kernel* (SSTK) was proposed in [14], where Predicate Argument Structures (PAS) are exploited to take dependencies into account. However, it was noted to be computational expensive for real world applications.

In the remainder of this Section, we introduce a new retrieval model, named *Syntactic Tree Matching (STM)*, by reformulating the original tree kernel definition. We present a new weighting scheme for tree fragments to make the final distance metrics not only faithful to the similarity measure but robust enough against some grammatical errors. This gives rise in Section 4 to a fuzzy matching scheme, which incorporates semantic features and elegantly tackles the second limitation.

3.1 Weighting Scheme of Tree Fragments

We directly employ the definition of the tree fragment from [19], where terminal nodes were included as a part of tree fragments. Before introducing the weighting scheme of the tree fragment, we first give definition to the node weighing factor:

Preliminary 1: *The weighting factor δ_i denotes the importance of node i in the parsing tree. Its value differs for different types of nodes:*

- *δ_i=1.2, where node i is either the POS tag VB or NN*[1]
- *δ_i=1.1, where node i is either VP or NP*
- *δ_i=1 for all other types of nodes*

We believe that different parts of the sentence have different importance, and the nouns and verbs are considered to be more important than other types of terms such as article, adjective or adverb. We also boost up the nodes of verb and noun phrases, to show their higher priority over other ordinary ones.

With node weighing factor, we define the weighing coefficient (θ) of the tree fragment as follows:

Preliminary 2: *The weighting coefficient θ_k for tree fragment k conveys the importance of the tree fragment, whose value is the production of all weighing factors of node i that belongs to the tree fragment k, i.e.,* $\theta_k = \prod_{i \in fragment\ k} \delta_i$ (3)

Intuitively, if a tree fragment contains lots of important nodes, its importance would be higher, and vice versa. The weighing coefficient can be reformulated into a recursive function $\theta_k = \delta_k \prod_{j \in fragment\ k} \theta_j$, in which δ_k represents the weighing factor of the tree fragment root, and θ_j is the weighting coefficient brought from its sub-trees that directly connect to the root.

We further define the size of the sub-tree (S_i) and its weighing factor (λ), together with the depth of the sub-tree (D_i) and its weighing factor (μ) as follows:

Preliminary 3: *The size of the tree fragment S_i is defined by the number of nodes that it contains. The size of weighing factor λ is a tuning parameter indicating the importance of the size factor.*

Preliminary 4: *The depth of the tree fragment D_i is defined as the level of the tree fragment root in the entire syntactic parsing tree, with D_{root}=1. The depth weighing factor μ is a tuning parameter indicating the importance of the depth factor.*

The introduction of the size and depth factors of the tree fragment is to account for the fact that sub-trees with different sizes and at different levels have different impact on the whole parsing tree. This impact could be interpreted in two aspects. First, a larger tree fragment contains more variety of senses. If a large portion of two parsing trees are of the same, their similarity would be higher. Second, the tree fragments at the bottom levels carry more significant semantic information than those at the upper level. This is because nodes at the upper layer usually determine the surface structure of a whole sentence, whereas nodes at the bottom layer contain information like word sense, inner phrase structures, and chunk relations etc., which are a lot more crucial.

The two tuning parameters λ and μ denote the preference between size and depth. Higher λ but lower μ means the size factor is more favorable than the depth factor, and vice versa.

Given the parameters listed above, we introduce the weighting scheme for the tree fragments:

Definition 1: *The weight of a tree fragment w_i is defined as $\theta_i\lambda^{Si}\mu^{Di}$, where θ_i is its weighting coefficient, S_i is the size of the sub-tree, λ is the size weighing factor, D_i is the depth of the sub-tree and μ is the depth weighting factor.*

Different from the tree weighting in [5], which penalized larger trees, our weighting scheme favors larger trees. Unlike the weighting proposed in [19], which simply considers the size and depth of the tree, our weighting scheme additionally takes into account the importance of the words or phrases that a tree fragment covers.

3.2 Measuring Node Matching Score

After introducing the weighting scheme of tree fragments, we need to match tree fragments and compute weights of the matched trees:

Preliminary 5: *If two tree fragments TF_1 and TF_2 are identical, the weight of their resulting matching tree fragment TF is defined to be* $w(TF) = w(TF_1)w(TF_2)$. (4)

Recall that the weighting scheme of each tree fragment is determined by the formula $\theta_i\lambda^{Si}\mu^{Di}$, we may thus write the weight of the matched tree fragment as $\theta_1\theta_2\lambda^{S1+S2}\mu^{D1+D2}$.

In view of the above, we introduce a new scoring function, named *node matching score*, between two nodes r_1 and r_2:

Preliminary 6: *The node matching score between two nodes r_1 and r_2 is the multiplication of weights of all matched tree fragments under the roots of r_1 and r_2. We use the following formula to describe it:*

$$M(r_1, r_2) = \begin{cases} 0 & \text{if } r_1 \neq r_2 \\ \prod_{i=1}^{\eta} w(TF_i(r_1, r_2)) & \text{otherwise} \end{cases} \tag{5}$$

where $r_1 \neq r_2$ denotes the fact that either labels or production rules for r_1 and r_2 are different, $TF_i(r_1,r_2)$ is the *i-th* matching tree fragment under r_1 and r_2, and η is the total number of tree fragments.

We can reformulate the *node matching score* into following recursive version:

$$M(r_1, r_2) = \begin{cases} \delta_{r1}\delta_{r2}\lambda^{S_1+S_2}\mu^{D_1+D_2}, \text{if } \mathrm{r}_1, \mathrm{r}_2 \text{ are terminals} \\ \\ \delta_{r1}^{\eta}\delta_{r2}^{\eta}\lambda^{2\eta}\mu^{\eta[2-(1+nc(r1))(D_{r1}+D_{r2})]} \\ \times \prod_{j=1}^{nc(r_1)} M(ch(n_1, j), ch(n_2, j)), \text{otherwise} \end{cases} \tag{6}$$

where *nc(n)* is the total number of children of the node *n*, *ch(n,j)* is the *j-th* child of node *n* in the tree, and η is the total number of matched tree fragments (See Appendix A for proof of correctness of the recursive function 6).

According to the comprehensive definition of weighting scheme for the tree fragments, two nodes with many tree fragments of higher weights are likely to produce higher node matching scores. This indicates that these node pairs may have covered very similar phrases. Therefore, we argue that the node matching score provides a good measure of the similarity between the sub-trees rooted under nodes r_1 and r_2.

3.3 Similarity Metrics

In order to find the similarity score between two syntactic parsing trees T_1 and T_2, we traverse them in post-order, and calculate the pair-wise *node matching scores* between the nodes in these two trees. This results in a $|T_1|$x$|T_2|$ matrix of $M(r_1,r_2)$. We

[1] Due to stemming, we normalize all POS tags in the way that all plural noun POS tags are replaced by their single forms (e.g. *NNS→NN*) and all verb POS tags are replaced by their base forms (e.g. *VBN→VB*).

use the summation of all scores in the matrix to represent the similarity score between two parsing trees:

Definition 2: *The similarity score or the distance metrics between two parsing trees is defined as*

$$sim(T_1,T_2)=\sum_{r_1\in T}\sum_{r_2\in T}M(r_1,r_2) \quad (7)$$

However, as the score is very sensitive to the size of trees T_1 and T_2, we normalize it into the following:

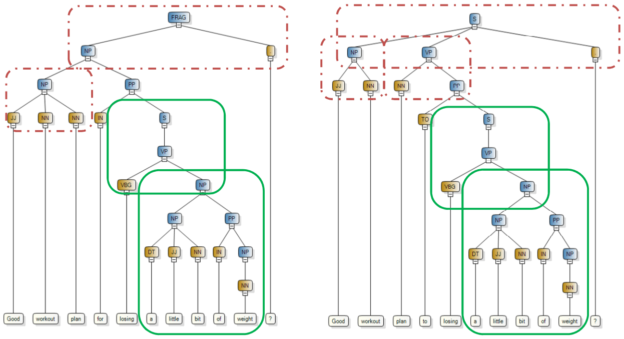
$$sim2(T_1,T_2)=sim(T_1,T_2)/\sqrt{sim(T_1,T_1)sim(T_2,T_2)} \quad (8)$$

By making use of the recursive definition of the node matching score, one can calculate the final similarity score between two parsing trees in polynomial time with dynamic programming.

3.4 Robustness

In real world, however, grammatical and spelling errors made by people are not uncommon, and these errors may have various influences on the resulting parsing tree. We observe that some of them such as article errors, tense errors, plurality errors etc. only affect a small portion of the parsing tree at the deep level. For instance, the parsing trees for the sentences "*I want doctor.*" and "*I want a doctor."* differ only by one leaf node and one POS tag (*DT*). We name these errors with only minor effects on the parsing result at the deep level as *interior errors.* Our matching model is obviously safe for them.

However, there are some other grammatical errors which may greatly alter the appearance of the parsing tree. We name them as *exterior errors,* due to the fact that they may change the shallow structure of the parsing tree. The preposition error, for example, is a kind of exterior errors. Figure 2 shows two parsing trees for two questions, in which 2(b) uses the preposition "*to*" instead of "*for*".



a) Parsing Tree for the sentence (with no grammatical error): *"Good workout plan for losing a little bit of weight?"*

b) Parsing Tree for the sentence (with grammatical error): *"Good workout plan to losing a little bit of weight?"*

Figure 2. Example on Robustness by Weighting Scheme

As has been spotted by the dashed rectangles in Figure 2, the surface structures of these two parsing trees appear to be very different due to the preposition misuse. However, it is also observed that, the structures of the chunks or phrases at the lower level are well preserved (as highlighted by the solid rectangles). This is common, as our large numbers of investigations show that the tree fragments at the lower level can be immune from exterior errors. In respect that tree fragments at the lower level are not affected by the exterior errors and the weight of the tree fragments at the lower level is relatively higher than those at the upper level, the matching score between two parsing trees will not be degraded much in case of errors. We therefore claim that our weighting scheme is robust to exterior grammatical errors.

4. SEMANTIC-SMOOTHED MATCHING

In the STM model above, if two parsing trees employ different leaf wordings or slightly transformed production rules, the tree fragments can hardly be matched. This becomes an evident drawback from the semantic point of view, and it motivates a modification to our original matching model. In order to capture more semantic meanings, we: (a) allow partial contribution from terminal words if they are shown to be closely related; (b) relax the production rules to allow for partial matching; and (c) use answer matching to bring in more semantically related questions.

Firstly, we use WordNet, a freely available semantic network, to help measure the semantic similarity between two words. We employ Leacock's measure [12], which uses the distance of the shortest path between two synsets to represent the semantic distance between two words, where the value is scaled by the overall depth of the taxonomy. In order to fit our matching model, in which the semantic score needs to be scaled between 0 and 1, we modify the Leacock's measure into the following:

$$Sem(w_1,w_2) = 1- distance(w_1,w_2)/2D \quad (9)$$

where $distance(w_1,w_2)$ is the length of the shortest path between two synsets of w_1 and w_2, and D is the maximum depth of the taxonomy. In particular, we define the path length between two identical words to be 0, i.e., $distance(w,w)$=0, or $Sem(w,w)$=1.

Secondly, we allow partial matching of production rules in the way that two nodes with sufficiently similar production rules can be matched. This sufficiency includes omission or reversion of the modifiers, preposition phrases, conjunctions and so on. For instance, "*NP→DT·JJ·NN*" is considered to be similar to "*NP→DT·NN*", and can be matched. The complete matching rules are not listed here due to space.

With the two relaxations defined above, we perform fuzzy matching between tree fragments. This could be achieved by modifying the matching scheme as proposed in *Preliminary 5:*

Preliminary 5': *The weight of the matching tree fragment TF resulted from matching TF_1* and *TF_2 is defined as:*

- $w(TF) = Sem(w_1, w_2)\delta_1\delta_2\lambda^{S_1+S_2}\mu^{D_1+D_2}$,

  if TF_1 and TF_2 are two terminal nodes w_1 and w_2;
- $w(TF) = \theta_1\theta_2\lambda^{S_1+S_2}\mu^{D_1+D_2}$,

  if the root of TF_1 and TF_2 are identical and their production rules can be partially matched.

The new definition is in line with *Preliminary 5*, except for the handling of terminal words and production rules. Two different terminal words can now be matched into a fuzzy word, and nodes with similar production rules can be aligned as well. $Sem(w_1,w_2)$ is the semantic similarity score as calculated from WordNet.

In order to avoid generating too many improbable tree fragments and make the matching more accurate, we impose two restrictions in our design:

1. A confidence level of 0.75 is set on semantic distances between two words. In other words, only two words with a sufficiently high semantic score could be matched.

2. Only terminal words with the same POS tag[2] could be matched. This prevents the matching of the word "*book*" in phrases like "*reading book*" and "*book air tickets*", which is obviously unreasonable.

It is noticed that question matching, even at the semantic level as described above, do not suffice to capture all similar pairs in some circumstances. For example, two similar questions "*Proper way to lose weight?*" and "*I'm too fat, help?*" hardly share any common points. To further overcome this kind of semantic gap, we introduce additional matching of questions via their answers, named *answer matching*. This matching is based on the intuition that if the answers to two questions are similar, the questions are considered to be semantically similar even if they are lexically very different. Therefore, given a query, the answers to top ranked matching questions could be utilized to fetch more similar questions via answer matching, where the newly retrieved questions could have great variations in both lexicon and syntax.

5. EXPERIMENTS

In this section, we present empirical evaluation results to assess the effectiveness of our STM technique for the similar question matching problem. In particular, we conduct experiments on the Yahoo! Answers QA archive and show that our STM is more effective than the original Tree Kernel function. We further show that the semantic-smoothed version gives additional boosting on matching precision.

5.1 Dataset

We issue *getByCategory* query provided in Yahoo! Answers API[3] calls to download QA threads from the Yahoo! site. We collected a total of around 0.5 million QA pairs from the *Healthcare* domain, over a 10-month period from *15/02/08* to *20/12/08*. It covers areas including diet, fitness, dental, diseases, men's and women's health, etc. We only focus on all *resolved* QA pairs, meaning questions that already have been given their best answers. Based on the hypothesis that the best answer represents the most accurate information responding to the question, we can use it to directly answer a query should we find a similar question to the query.

As there can be multiple questions asked in a single question thread, we segment each question thread into pieces of single-sentence questions by using question mark and 5W1H words heuristic. The reason is two-fold: (a) Different questions may ask about different aspects; to separate them is helpful to better match questions with user's query. (b) The parser handles short sentences better than longer ones, for which ambiguous syntactic structures are likely to occur.

In order to evaluate our retrieval system, we divide our dataset into two parts. The first part (0.3M), covering the initial period of 3.5 months dated from 15/02/08 to 05/06/08, is used as the ground-truth setup; the rest is used as test-bed for evaluation. For ground-truth, we asked four annotators to tag similar questions from the first part of the dataset. As the number of question threads is huge, it becomes infeasible for annotators to go through all to check their similarities. To ease this, we employ the K-means text clustering method to first group similar answers. The rationale behind this is based on the assumption that two questions are considered to be similar if their answers are similar. The answer groups thus help to find corresponding similar questions[4]. Among these clusters, we diversely choose 20 representative groups for each sub-category, in order to ensure well coverage on topics in each domain. A series of simple BoW-based retrievals are then performed on each group to get in more potentially similar questions. We believe the resulted question groups, which have potentially covered both lexical and semantic similarity, are the good starting point for the tagging task.

The tagging results, together with the dataset statistics, are shown in Table 1. There is a total of 301,923 question threads from 6 sub-categories and on average 1.96 questions were asked per question thread (referred to as "*Q Ratio"*). Among all, annotators have tagged 120 (20x6) groups of similar questions, with a total of 10255 questions serving as the ground-truth for later evaluation.

Table 1. Statistics of Dataset Collected for the Ground-Truth

Category	# of Question Thread	Est # of Questions	Q Ratio	# of Ground Truth
Dental	28879	59349	2.06	875
Diet&Fitness	105079	202331	1.93	5905
Diseases	31017	59259	1.91	454
General Healthcare	23004	45067	1.95	1008
Men's Health	42017	77342	1.84	793
Women's Health	71930	149880	2.08	1220
Total	301923	593228	(avg) 1.96	10255

Each annotator was also asked to indicate the topic of each group of similar questions. We use these topics as a guidance to choose the testing questions for our evaluation. A total of 120 questions, which are considered to be close enough to its groups in the ground-truth, are carefully chosen from the testing set for testing. These questions are of various lengths and in various forms. Table 2 shows some example queries from this testing set.

Table 2. Example Queries from Testing Set

Query	**Category**	**Topic**
What is the best way to use crest white strips premium plus?	Dental	Whitening Strips
Tips on losing weight?	Diet & Fitness	Weight loss
Tingling in legs, sometimes pain, what is it?	Diseases	Pain in legs
Why is it that at the same time afternoon or night I always go tired?	General Healthcare	Feeling tired
Any advice on a fitness schedule including weight lifting and diet plan?	Men's Health	Advice on fitness

5.2 Retrieval Model

We first index all the collected questions and answers from Yahoo! Answers. By given a user query, an initial BoW retrieval is carried out on question index, where different retrieval techniques such as term weighing and relevance feedback are applied. Top 100 of the initial retrieval results (R_STM) are then selected, each of which is matched against the user query via the STM module. A re-ranked matching result is then produced. We

[2] Currently we focus on NN and VB matching, as WordNet only provides the hypernymy hierarchical relationship for nouns and verbs. For adjectives and adverbs, we may look into their synonyms, but it is difficult to give a quantitative similarity score between them.

[3] http://developer.yahoo.com/answers/

[4] In order to generate clusters with higher inner similarity, we set a very high threshold to filter out irrelevant ones.

further perform the answer matching to bring in more similar questions (R_AM). Two sets of questions are fused with linear interpolation ($\alpha \times R\_STM + (1-\alpha) \times R\_AM$) to make up the final similar question searching result. Figure 3 presents an overview of our retrieval system.

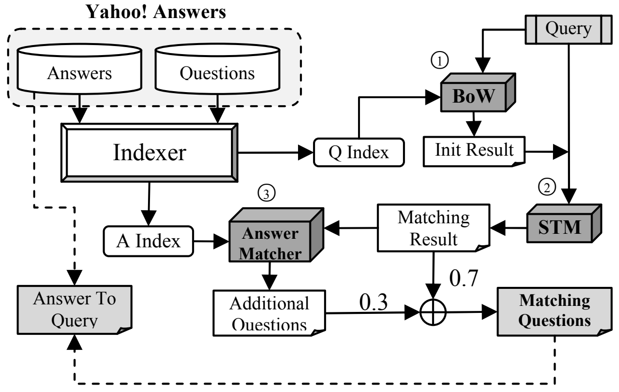


Figure 3. Overview of Question Matching System

5.3 Performance Evaluation

To evaluate the performance of our retrieval model, we use five different system combinations for comparison:

1) BoW (baseline1): A Bag-of-Word approach that simply matches stemmed words between the query and questions.
2) BoW+TK (baseline2): BoW integrated with the original tree kernel function (sub-tree counting) for question matching.
3) BoW+STM: BoW approach combined with the Syntactic Tree Matching model as introduced in this paper.
4) BoW+STM+SEM: Matching model 3) with semantic features incorporated.
5) BoW+STM+SEM+AM: Matching model 4) with answer matcher module integrated.

Table 3. MAP Performance on Different System Combinations and Precision at Top 1 Retrieval Results

System Combination	BoW	BoW+TK	BoW +STM	BoW+STM +SEM	BoW+STM +SEM+AM
MAP (%)	79.08	81.61	85.67	86.41	88.56
% improvement of MAP over:					
BoW	N.A.	+3.20	+8.33	+9.26	+11.99
BoW+TK	N.A.	N.A.	+4.97	+5.88	+8.51
Precision at Top 1	81.67	82.50	81.67	88.33	89.17

We employ two performance metrics: mean average precision (MAP_{10}[5]) and precision at the top one retrieval result. The evaluation results are illustrated in Table 3. From the Table, we draw the following observations:

1) BoW model itself achieves very high precision (79.08), and BoW+TK slightly improves BoW by 3.20%. We conjecture that the high precision obtained by BoW is because of the huge size of the Yahoo! Answers archive, where a large number of lexically similar question threads have already been stored. Users are therefore quite likely to get similar questions even using some key words. The combination of the TK and BoW gives very limited boosting, leading us to be more convinced that TK does not capture the similarity between questions well.
2) Applying syntactic tree matching over simple lexical matching methods boosts system performance a lot. When applied on top of BoW, both STM and SEM augment the performance in all metrics statistically significantly as judged by using paired t-test (p-value<0.01). MAP on BoW+STM improves by 8.33% and 4.97% respectively over the results obtained by BoW and BoW+TK; and the MAP is improved by 9.26% and 5.88% respectively when the semantic features are incorporated. Semantic-smoothed syntactic tree matching also yields better precision in the top one retrieval task, where it retrieves questions correctly at the first position on 106 questions out of a total of 120. We believe that the improvement stems from the ability of the syntactic match weighting scheme to correctly present and measure the similarity distance between questions. As such, many false positive questions that would be favored by normal BoW approaches are subsequently eliminated, as they often do not contain similar syntactic structure with the user query.
3) Semantic-smoothed matching performs better than pure syntactic matching marginally. It gains an improvement of 0.74% in MAP and 6.66% in precision at top one question result when using STM+SEM over STM itself. We conjecture that the marginal improvement of semantic features in terms of MAP is probably because there are too many lexically similar questions existing in the Yahoo! Answers archive. In other words, the performance of the baseline system is so high that little room is left for further contribution by the use of semantic features. We believe that the semantic-smoothed matching may give significant improvement over others in the environment where there is a large number of semantically similar but lexically different terms or phrases, as it may accommodate the variation in natural language texts, for which pure syntactic matching may fail to capture.
4) Answer matching (AM) brings in significant improvements over others in all metrics. This is in line with our expectation. By the hypothesis that similar questions give similar answers, the AM module is capable of bringing in more semantically similar questions which bear totally different words or structures. It is analogous to relevance feedback module as in traditional IR systems, where new query can be formed according to the initial retrieval, and new results can be obtained.

5.4 Performance Variations to Grammatical Errors

To support our statement in Section 3.4 that the STM is robust to grammatical errors, we conduct experiments in this Section to examine the effect of various grammatical errors on MAP.

For meaningful comparison, we mimic a noisy environment by manually injecting various common errors that the human users are likely to make into all testing questions. 50% of the testing questions are randomly inserted with *interior errors* like article, tense, plurality errors, while the other 50% are modified to bear random *exterior errors,* such as preposition errors, misplaced

[5] $\text{MAP}_{10:}$ The MAP calculated on the returned top 10 questions.

modifier etc. Along with grammatical errors in our simulation, we also consider modifying some terms or phrases into non-standard short forms that have been frequently used on the Web, such as "*4*" for "*for*", "*your*" for "*you are*", and "*Im*" for "*I am*" etc.

We re-run the five systems on the modified testing dataset and plot the results of performance variations in Figure 4. The top of bars indicates the original performance without grammatical errors and the bottom shows the resulting performance.

We can see from Figure 4(a) that the systems with STM integrated still outperform the BoW and TK systems by a large margin in terms of MAP even in noisy environment. The top one precision of STM-embedded systems does not degrade as much in general as compared to TK as shown in Figure 4(b). In fact, the BoW+STM system even outperforms the BoW and BoW+TK systems in noisy environment. This is evidence that the syntactic tree matching model is sufficiently robust to various forms of grammatical errors. We expect that in real world situation, a STM-based system would give very satisfying matching results.

Interestingly, the MAP for BoW-only system does not drop much as compared to the others. We believe this is owing to the fact that purely lexical based approaches do not take word relations into consideration, and thus it is less influenced by the grammatical errors.

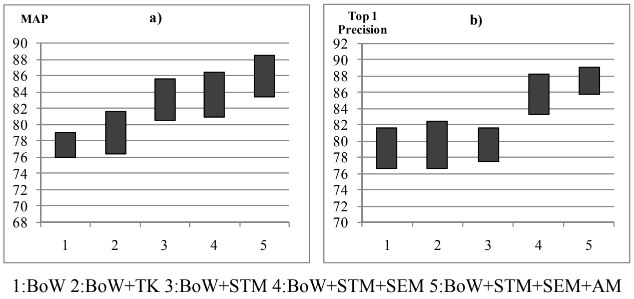


Figure 4. Illustration of Variations on a) MAP b) Top one Precision to Grammatical Errors.

5.5 Error Analysis

Although we have shown that STM, together with SEM and AM improves question matching, there is still plenty of room for improvement. To further characterize the types of questions that have no impact or are adversely affected by STM, we perform micro-level error analysis on the testing question set. We find that STM fails to match the syntactic structures of questions mainly due to the following three reasons:

1) Mismatch of question topics: In some cases, two questions asking about different topics but bearing very close sentence structures are incorrectly matched. For instance, the question "*How do I increase my appetite in a short period?*" is highly ranked by given the query "*How can I increase my height in a few month?*". This becomes severe for short questions asking about different aspects, as simple sentence structures are likely to have exactly the same syntactic structure as the others. To overcome this problem, we need to incorporate question analysis in our system such that the question target could be clearly identified and compared when performing the matching.

2) Flexibility of question representations: In real world, many semantically similar questions have vast differences in their expressions. For example, questions like "*Best ways and products to get fresh breath?*", "*How do you keep your breath always smelling fresh and clean?*" and "*Has anyone tried bad breath cures?*" etc. are tagged to be similar in the ground-truth with the topic of "bad breath remedies". However, the query "*Please tell what I can do to make my mouth smell go away?*" in the testing dataset matches none of these. Both lexical matching and syntactical matching are likely to fail in this case as we are lack of not only ways of correlating different human expressions, but also information on the relation between the term "breath" and the phrase "mouth smell". This is a difficult challenge, and we believe that certain domain ontology could be of help, where at least some semantically related terms can be linked together.

3) Extremely long queries: There are also some cases that the query question is too long for STM to give correct similarity measure. We conjecture that the syntactic structure of a sentence becomes more complicated when the sentence gets longer, and this leads to higher flexibility of interchanging different sentence components, resulting in a large number of nodes at the upper layers to be mismatched and the failure of the STM measure. There are some works [6,8] that attempt to find dependency relations between terms to counterbalance the word ordering problem. However, their method is also limited to short queries, as the dependency parser tends to perform worse for longer sentences.

6. RELATED WORK

The idea of finding similar questions in cQA is to some extent related to passage retrieval in traditional QA, with the exception that question-to-question matching is much stricter than question-to-passage matching. Many techniques have been developed towards passage retrieval, from the simple BoW-based model, language model [16], to some state-of-the-art techniques like dependency relations [6] etc. Most of these techniques can be employed to match questions, but their precision is not high because of the high recall requirement. Training is needed in some works as well.

Likewise, the FAQ retrieval task is also closely related to the question matching problem. Early work such as the FAQ finder [4] combined statistical similarity measure with semantic measure using WordNet to rank FAQs. Some recent works [15,17] used more advanced translation-based approaches to retrieve FAQ data. [11] and [10] mined the FAQ data from the Web and implemented their own retrieval systems.

However, the community-based QA archive is different from FAQ collections in the sense that the scope of manually created FAQs is quite limited. Recent research begins to focus on large scale QA services from the Web. Some works have been conducted on the characteristic analysis on this type of services such as [2], but limited effort has been devoted to the question matching direction. Works proposed in [9] and [18] applied the translation-based model to find semantically similar questions in cQA.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel syntactic tree matching method for the similar questions finding problem. We assessed the system based on the ground-truth built from Yahoo! Answers, and the evaluation results showed that our system produced competitive improvements in matching performance as compared to the traditional BoW or plain tree kernel function: a 5~12% improvement in MAP, and up to 8% in top one precision. We introduced a comprehensive tree weighting scheme to not only give a faithful measure on question similarity, but also handle grammatical errors gracefully. We further improved the system performance by incorporating semantic features and the answer matching module. Unlike other systems, our model does not rely on training, making it easily portable to other similar retrieval systems.

Our empirical evaluation results and qualitative error analysis revealed that the syntactic tree matching model could be improved by integrating question analysis module and domain ontology. Moreover, most off-the-shelf parsers, including the one we used in our experiments, are not well-trained to parse questions. We believe that a more targeted parser which is trained on question sets may give better accuracy.

The retrieval system in this work only focuses on the single-sentence question matching problem, and uses the best answers as it is. In future research, multiple-sentence questions with different purposes are to be analyzed, and a tailored answer summarization technique is to be developed as well to produce high quality answers from different sources.

8. REFERENCES

[1] Trec proceedings. http://trec.nist.gov/proceedings/proceedings.html.

[2] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *WWW '08*, pages 467–476. ACM, 2008.

[3] S. Bloehdorn and A. Moschitti. Structure and semantics for expressive text kernels. In *CIKM '07*, pages 861–864. ACM, 2007.

[4] R. D. Burke, K. J. Hammond, V. A. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI Magazine*, 18(2):57–66, 1997.

[5] M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632. MIT Press, 2001.

[6] H. Cui, R. Sun, K. Li, M.-Y. Kan, and T.-S. Chua. Question answering passage retrieval using dependency relations. In *SIGIR '05*, pages 400–407. ACM, 2005.

[7] A. Diekema, X. Liu, J. Chen, H.Wang, N. Mccracken, O. Yilmazel, and E. D. Liddy. Question answering: Cnlp at the trec-9 question answering track. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 501–510. Department of Commerce, National Institute of Standards and Technology, 2000.

[8] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. Dependence language model for information retrieval. In *SIGIR '04*, pages 170–177. ACM, 2004.

[9] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *CIKM '05*, pages 84–90. ACM, 2005.

[10] V. Jijkoun and M. de Rijke. Retrieving answers from frequently asked questions pages on the web. In *CIKM '05*, pages 76–83. ACM, 2005.

[11] Y.S. Lai, K.A. Fung, and C.-H. Wu. Faq mining via list detection. In *COLING-02*, pages 1–7. Association for Computational Linguistics, 2002.

[12] C. Leacock and M. Chodrow. Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[13] A. Moschitti. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, pages 318–329. Springer, 2006.

[14] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. Exploiting syntactic and shallow semantic kernels for question answer classification. In *ACL*. The Association for Computer Linguistics, 2007.

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

[16] F. Song and W. B. Croft. A general language model for information retrieval. In *CIKM '99*, pages 316–321. ACM, 1999.

[17] R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In *HLT-NAACL*, pages 57–64, 2004.

[18] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *SIGIR '08*, pages 475–482. ACM, 2008.

[19] D. Zhang and W. S. Lee. Question classification using support vector machines. In *SIGIR '03*, pages 26–32. ACM, 2003.

Appendix: Proof of Recursive Function *M(r₁,r₂)*

By the definition of the *node matching score*, we have:

$$M(r_1,r_2)=\delta^{\eta}_{r1}\delta^{\eta}_{r2}\lambda^{2\eta}\mu^{\eta[2-(1+nc(r1))(D_{r1}+D_{r2})]}\prod_{j=1}^{nc(r_1)}M(ch(n_1,j),ch(n_2,j))$$

$$=\delta^{\eta}_{r1}\delta^{\eta}_{r2}\lambda^{2\eta}\mu^{\eta[2-(1+nc(r1))(D_{r1}+D_{r2})]}\prod_{j=1}^{nc(r_1)}\prod_{i=1}^{\eta'}w(TF_i(ch(n_1,j),ch(n_2,j)))$$

$$=\delta^{\eta}_{r1}\delta^{\eta}_{r2}\lambda^{2\eta}\mu^{\eta[2-(1+nc(r1))(D_{r1}+D_{r2})]}\prod_{j=1}^{nc(r_1)}\prod_{i=1}^{\eta'}\theta_{i1}\theta_{i2}\lambda^{S_{i1}+S_{i2}}\mu^{D_{i1}+D_{i2}}$$

$$=\prod_{k=1}^{\eta}\left[\delta_{r_1}\delta_{r_2}\lambda^{2}\mu^{2-(1+nc(r1))(D_{r1}+D_{r2})}\prod_{j=1}^{nc(r_1)}\theta_{k1}\theta_{k2}\lambda^{S_{k1}+S_{k2}}\mu^{D_{k1}+D_{k2}}\right]$$

$$=\prod_{k=1}^{\eta}\left[\begin{matrix}\left(\delta_{r_1}\prod_{j=1}^{nc(r_1)}\theta_{j1}\right)\left(\delta_{r_2}\prod_{j=1}^{nc(r_1)}\theta_{j2}\right)\left(\lambda^{1+\sum_k S_{k1}}\right)\left(\lambda^{1+\sum_k S_{k2}}\right)\\ \mu^{2-(1+nc(r1))(D_{r1}+D_{r2})+nc(r1)(D_{k1}+D_{k2})}\end{matrix}\right]$$

As the weighting coefficient θ has the recursive definition of $\theta_r=\delta_r\prod_{j=1}^{nc(r)}\theta_j$, the size of the new tree fragment has the relations of $S_r=1+\sum_{k\in Children(r)}S_k$, and the depth of the new tree fragment is just one level above its children's, which gives $D_r = D_k - 1$, we may finally get the following:

$$M(r_1,r_2)=\prod_{k=1}^{\eta}\left(\theta_{r_1}\theta_{r_2}\lambda^{S_{r1}}\lambda^{S_{r2}}\mu^{D_{r1}}\mu^{D_{r2}}\right)=\prod_{k=1}^{\eta}w(TF_k(r_1,r_2))\quad QED.$$