Structured Use of External Knowledge for Event-based Open Domain Question Answering

Hui Yang, Tat-Seng Chua, Shuguang Wang, Chun-Keat Koh
National University of Singapore
3 Science Drive 2, Singapore 117543
{yangh,chuats,wangshug,kohchunk}@comp.nus.edu.sg

ABSTRACT
One of the major problems in question answering (QA) is that the queries are either too brief or often do not contain most relevant terms in the target corpus. In order to overcome this problem, our earlier work integrates external knowledge extracted from the Web and WordNet to perform Event-based QA on the TREC-11 task. This paper extends our approach to perform event-based QA by uncovering the structure within the external knowledge. The knowledge structure loosely models different facets of QA events, and is used in conjunction with successive constraint relaxation algorithm to achieve effective QA. Our results obtained on TREC-11 QA corpus indicate that the new approach is more effective and able to attain a confidence-weighted score of above 80%.

Categories and Subject Descriptors
I.2.7 [**Natural Language Processing**]:– *Language parsing and understanding, Text analysis*.

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
Question Answering, Query Formulation, Event-based QA

1. INTRODUCTION
Most users are interested in searching for *information*, while the current search engines are designed to retrieve only *documents*. There are many simple factoid questions like: "*Who is Tom Cruise married to?*" or "*How many chromosomes does a human zygote have?*" While the users expect short precise answers to such questions, typical search engines would return thousands of documents where the actual answer is embedded in some of the documents. This has resulted in a gulf between the expectation of the users and that retrievable by the systems. This gulf will become more severe as we are increasingly being overwhelmed by information overloads.

To address this problem, a Question-Answering (QA) task was initiated in TREC conference series [16]. This has in turn motivated much of the recent research on open domain QA focusing on short, factoid questions. Most modern QA systems combine the strengths of traditional Information Retrieval (IR), natural language processing (NLP) and information extraction (IE) to provide an appropriate way to retrieve concise answers for open-domain natural language questions against a large text collection (termed the QA corpus). In the most recent TREC-11 conference [16], the main task consists of 500 short factoid questions posed over more than one million news articles. Instead of previous years' 50-byte or 250-byte text fragments, exact answers are expected from the QA corpus with supports of documentary evidences.

Although the traditional "bag-of-words" representation has been found to be effective for IR research in retrieving large number of relevant documents, it is ineffective for QA where precise answers are needed. Many QA research groups employ a variety of linguistic resources, such as the part-of-speech tagging, syntactic parsing, pattern matching, semantic relations, named entity extraction, dictionaries, WordNet, etc [7,8,10,12,13]. In the TREC-11 evaluation [16], Moldovan et al [11] successfully illustrated the power of extensive WordNet to perform logic proof relied on knowledge reasoning. In contrast to the linguistic-based approaches, the use of the Web for QA [3,4,14,18,19] is an emerging topic of interest among the computational linguistic communities [1,2,16]. Several studies suggested that using the Web as additional knowledge resource could improve the performance of a QA system by 25-30%. Clarke et al [4] and Brill et al [3] used the Web to introduce data redundancy for more reliable QA. Lin [8] gave a detailed comparison of employing the Web and WordNet and concluded that combining both approaches could lead to better performance on answering definition questions.

In TREC-11, we explored the use of external resources like the Web and WordNet to extract terms that are highly correlated with the query, and use them to perform linear query expansion [18]. While the technique has been found to be effective, we found that there is a need to perform structured analysis on the knowledge obtained from the Web/WordNet in order to further improve the retrieval performance. In this work, we model the TREC-style QA task as *QA entities or Events*. Each QA event comprises of elements describing different facets of the event like *time, location, object, action* etc. For most QA events, there are inherent associations among their elements. Thus if we know a portion of the elements, we can use this information to narrow the search for the rest of unknown event elements, which are likely to contain the answers. In order to incorporate more knowledge of event elements from the external resources and to use event structure systematically for more effective QA, we analyze the external knowledge to discover the event structure.

This paper investigates the integration and structured use of both linguistic and web knowledge for the TREC-style QA, which we

SIGIR'03, July 28-August 1, 2003, Toronto, Canada.

call ***Event-based QA***. In particular, we describe a high performance question answering system called **QUALIFIER (QUestion Answering by LexIcal FabrIc and External Resources)** and analyze its effectiveness using the TREC-11 benchmark. Our results show that the Event-based approach, together with the use of *Successive Constraint Relaxation,* gives rise to an effective question answering system.

The rest of the paper is organized as follows. Section 2 presents the idea of *Event-based Question Answering*. Section 3 discusses the design and architecture of the system. Section 4 elaborates on both the linear and structured use of external resources for QA. Sections 5 and 6 investigate the use of successive constraint relaxation and answer extraction technique. Section 7 details the experimental results and Section 8 concludes the paper.

2. EVENT-BASED QA

Research groups working on QA employ various models of questions and techniques for locating the answers. Techniques being employed include the use of the Web, lexical resources, ontology and linguistic tools, etc. In this research, we propose a novel way to investigate the QA problem by ***Event-based Question Answering***.

The world consists of two basic types of things: ***entities*** and ***events*** and people often ask questions about them. Examples of questions are *"What is the democratic party symbol?" "What year was Alaska purchased?"*. From the definitions in WordNet, ***an entity*** is "*anything having an existence (living or nonliving)*" and ***an event*** is "*something that happens at a given place and time*". If we apply this taxonomy to QA task, questions can be considered as *"enquiries about either entities or events"*. Usually, the entity questions expect the entity properties or the entities themselves as the answers, such as the *definition questions*. More generally, questions often show great interests in several aspects/elements of events, namely *Location, Time, Subject, Object, Quantity, Description* and *Action,* etc. Here we give the formal definitions of *Event-based QA* as follows:

question : == event | event_element| entity| entity_property
event : == { event_element }
event_element : == time |location |subject |object |quantity
|description| action |others
entity : == object | subject
entity_property : == quantity | description | others

Table 1 shows the correspondences of the most common WH-question classes and QA event elements.

Table 1:Correspondence of WH-Questions & Event Elements

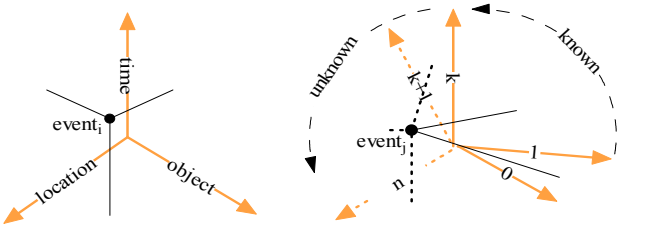
WH-Question	QA Event Elements
Who/Whose/Whom	Subject, Object
Where	Location
When	Time
What	Subject, Object, Description, Action
Which	Subject, Object
How	Quantity, Description

Besides the elements that are possible answers to a question, the question itself also contains some QA elements. Our main observation is that a QA event shows great cohesive affinity to all its elements and the elements are likely to be closely coupled by this event. Based on this observation, we derive the following ***Event-based QA Hypothesis***:

1) ***Equivalency***: *if all_elements(E_i) = all_elements(E_j), then E_i = E_j, and vice versa;*
2) ***Generality***: *if all_elements(E_i) is a subset of all_elements(E_j), then E_i is more general than E_j;*
3) ***Cohesiveness***: *if elements a, b both belong to an event E_i, and a, c do not belong to a known event, then co-occurrence(a,b) is greater than co-occurrence(a,c);*
4) ***Predictability***: *if elements a, b both belong to an event E_i, then a => b and b => a.*

(Here "=>" means "induces".)

We can consider an event to be a point within the multi-dimensional QA event space. In this space, if we know all the elements about an event, then we can easily answer different questions about this event (See Figure 1a for a 3-D case). Normally, the question itself provides *k* known elements, which are from the original query, and asks for some parts of the *(n-k)* unknown element(s), which will be the answer (See Figure 1b). As there are innate associations among these elements if they belong to the same event, we can use what are already known to narrow the search process to find the rest of unknown event elements, i.e., the answer. (Hypothesis 4)



a) Event elements for a 3-D event$_i$ (time, object and location) b) Knew *k* elements, looking for 1/more from *(n-k)* elements of high-D event$_j$

Figure 1: Illustration of QA Event Elements

However, for most of the cases, it is difficult to find a correct answer, i.e., the correct unknown element(s). There are two major problems: insufficient known elements *(k<<n)* and inexact known elements. To solve the first problem, we explore the use of world knowledge, which comes from the Web and WordNet glosses. To solve the second problem, we exploit the lexical knowledge from WordNet synsets and morphemics and employ a new technique called *Successive Constraint Relaxation* to successively remove the inexact elements that hinder the answer finding process. In addition, we need to perform structured semantic analysis based on what we already know in order to discover the event structure.

3. QUALIFIER SYSTEM ARCHITECTURE

Our system, named QUALIFIER, includes modules to perform question analysis, query formulation by using external resources, document retrieval, and exact answer extraction. The system architecture is shown in Figure 2.

The purpose of question analysis in QUALIFIER is to extract the known QA elements embedded in the question and the expected answer (or targeted unknown element) type. QUALIFIER extracts several sets of words (known elements) from the original question and employs a rule-based question classifier to identify the answer target (unknown element) type. Our rule-based question classifier adopts a two-tiered question taxonomy, which corresponds to fine-grained named entities (See Table 2 in Section 6) and achieves a classification accuracy of over 98% on recent year's TREC questions [18].

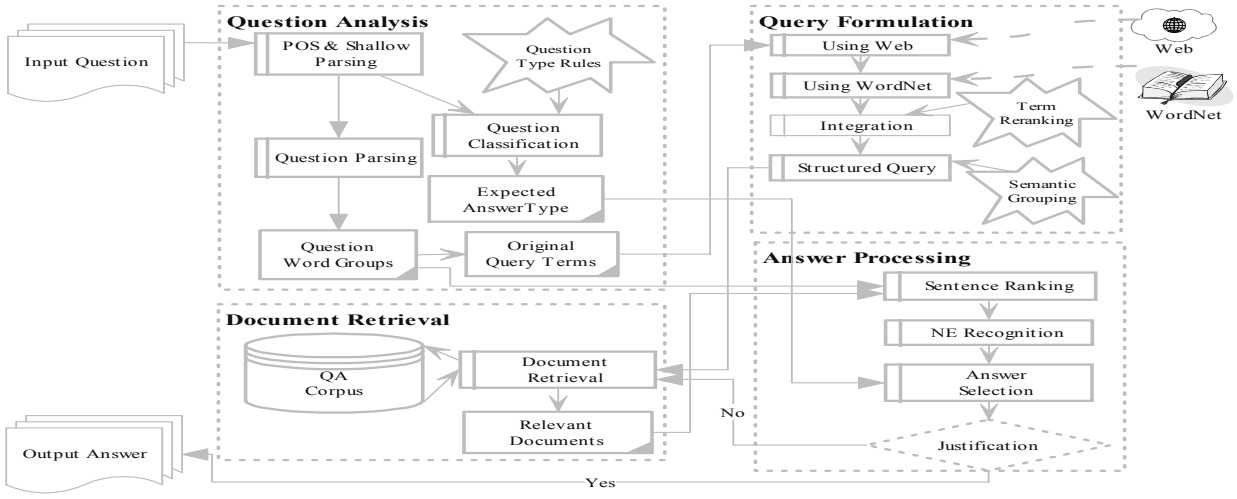


Figure 2: QUALIFIER System Architecture

During query formulation, QUALIFIER integrates the knowledge of both the Web and WordNet to extract context information for the query. We explore two ways to processing the external knowledge. One is the linear approach of extracting highly correlated terms from the relevant information derived from the Web/WordNet [19]. The other approach is to perform structured analysis to discover different facets or elements of events, and employ a structured (event-based) approach to perform query formulation. In this paper, we will concentrate more on the event-based query expansion, which will be discussed in the next section.

Given the newly formulated query, QUALIFIER employs the MG tool [17] to perform Boolean search for top ranked documents in the QA corpus so that we can narrow down the scope of unknown elements. It then selects candidate answer sentences based on the similarity between the sentences and the query in terms of original/expanded query terms, noun phrases, and quotations, etc [18]. Finally it performs linguistic and named entity analysis on the candidate sentences, and picks a named entity string with high possibility to be the targeted unknown element as the exact answer. Successive Constraint Relaxation is used here to improve the answer coverage.

4. STRUCTURED USE OF EXTERNAL KNOWLEDGE

Given a short, fact-based question, it can be transformed to a query in a form of $\underline{q}^{(0)} = [q_1^{(0)}\ q_2^{(0)}\ \ldots q_k^{(0)}]$. The problem for retrieving all the documents relevant to $\underline{q}^{(0)}$ is that *the query does not contain most of the terms used in the document space to represent the same concept*, i.e., *the query does not provide sufficient and correct QA event elements for us to locate the targeted QA event.*

For example, given the question: *"What Spanish explorer discovered the Mississippi river?"*, the query $\underline{q}^{(0)}$ will be [*Spanish, explorer, Mississippi, river*]. Two of the passages containing possible answer in the QA corpus are:

a. *Pecans are native to the lower Mississippi Valley but not to Georgia. The 16th-century Spanish explorer Hernando De Soto, who found the Mississippi River, wrote about these nuts. They are a relative of the hickory, and their name comes from a Cree word meaning "bone shell" or "hard shell."*

b. *Historians think Hernando de Soto was Tampa Bay's first tourist, landing near modern Bradenton in 1539 and lingering only briefly before heading west on a fruitless search for jewels and gold that ended with his death on the banks of the Mississippi in 1542.*

As can be seen, there are few common content words between the question and the passages. If we attempt to retrieve possible answer passages by using the original query, we will get too many irrelevant passages. In order to constrain the query and to bridge the gap between the query and the QA corpus, we resort to using general open resources to overcome this problem. In our system, we focus on the amalgamation and structured use of the external knowledge like the Web and WordNet.

4.1 Integrating External Knowledge

The Web is the most rapidly growing and complete knowledge resource in the world. We expect that the terms used in the relevant documents retrieved from the Web are similar to or even the same as those in the QA corpus since they both contain same information about the natural facts *(QA Entity)* or the factual events in the history *(QA Event)*. Inspired by the *Cohesiveness hypothesis* of the Event-based QA (See Section 2), we use the Web to retrieve the relevant information for a certain entity or element(s) of an event based on term co-occurrence.

To achieve this, QUALIFIER uses the original content words in the question and keeps them in $\underline{q}^{(0)}$ to retrieve the top N_w documents using the Web search engine (E.g. Google), and extracts the terms in these documents that are highly correlated with the original query terms in the local context. That is, for $\forall q_i^{(0)} \in \underline{q}^{(0)}$, it extracts the list of nearby non-trivial terms, $\underline{t}_i$, that are within the same sentence or snippet as $q_i^{(0)}$. QUALIFIER further computes the weights for all terms $t_{ik} \in \underline{t}_i$ based on their probabilities of correlation with $q_i^{(0)}$ as:

$$weight(t_{ik}) = \frac{d_s(t_{ik} \wedge q_i^{(0)})}{d_s(t_{ik} \vee q_i^{(0)})} \quad (1)$$

where $d_s(t_{ik}/\!\backslash q_i^{(0)})$ gives the number of web snippets or sentences that contain both t_{ik} and $q_i^{(0)}$; and $d_s(t_{ik}\backslash\!/ q_i^{(0)})$ gives the number of

web snippets or sentences that contain either t_{ik} or $q_i^{(0)}$. Finally, QUALIFIER merges all $\underline{t}_i$ to form $\underline{C}_q$ for $\underline{q}^{(0)}$.

The Web is useful at finding the world knowledge by providing the words that occur frequently with the original query terms in the local context. However, it lacks information on lexical relationships among these terms, such as synonyms. To extract such knowledge, QUALIFIER employs WordNet to get the glosses and synsets of the original query terms in order to provide more useful terms related to the QA event.

However, we need to be careful when extracting the gloss words $\underline{G}_q$ and synset words $\underline{S}_q$ for $\underline{q}^{(0)}$ from WordNet. This is because for a QA event, each term normally manifests only one major sense. To ensure that we do not pick terms out of context, we: (a) form $\underline{S}_q$ by using the synset words from the first sense of the first part-of-speech of the terms; and (b) form $\underline{G}_q$ by either using all the gloss words of the specific terms which have only one or two senses, or just the frequent gloss words for general terms which have many senses. We then combine the external knowledge sources by adding the words in $\underline{C}_q$, and those in $\underline{G}_q$ and $\underline{S}_q$ to form $\underline{K}_q$:

$$\underline{K}_q = \underline{C}_q + (\underline{G}_q \cup \underline{S}_q) \quad (2)$$

At the end of this process, $\underline{K}_q$ contains many terms related to the known elements of a QA event. We have studied two possible approaches to investigate the potential of Event-based question answering. One simple approach is to treat each term in $\underline{K}_q$ as a list without any structure and another is to form semantic word groups before building the new query. We call them *Linear Query Formulation* and *Structured Query Formulation* respectively and will discuss in more details in the following subsections.

4.2 Formulating Linear Queries

When performing Linear Query Formulation, we simply treat the terms in $\underline{K}_q$ as a list and perform the query expansion. However, if we simply append all the terms, the resulting expanded query will be too broad and contain too many terms out of context. Hence, we have to restrict the glosses and synonyms to only those terms found in the web documents in order to ensure that they are in the right context. We tackle this problem by using $\underline{G}_q$ and $\underline{S}_q$ to increase the weights of terms found in $\underline{C}_q$ as follows:

Given term $t_k \in \underline{C}_q$:
- if $t_k \in \underline{G}_q$, increase weight for t_k by α;
- if $t_k \in \underline{S}_q$, increase weight for t_k by β;

where $0 < \beta < \alpha < 1$.

The final weight of each term is normalized and the top *m* terms above the cut-off threshold σ are selected to expand the original query:

$$\underline{q}^{(1)} = \underline{q}^{(0)} + \{\text{top } m \text{ terms} \in \underline{C}_q \text{ with weights} \geq \sigma\} \quad (3)$$

where *m* is initially set to 20 in our experiments.

The expanded query should contain many more known elements than the original query and hence we have more overlapping terms or concepts with the passages containing the answers in the QA corpus. For the "*Mississippi*" example, the original query terms are *"Spanish, explorer, Mississippi, river"*. After we employ the Web, we get the expanded query *"Mississippi, Spanish, river, explorer, Hernando, Soto, De, 1541, first, European, French"*. The WordNet is then used to re-rank the query terms, and the resulting query ranked list is: *"Mississippi, Spanish, Hernando, Soto, De, 1541, Explorer, First, European, French, river"*. (See Figure 3)

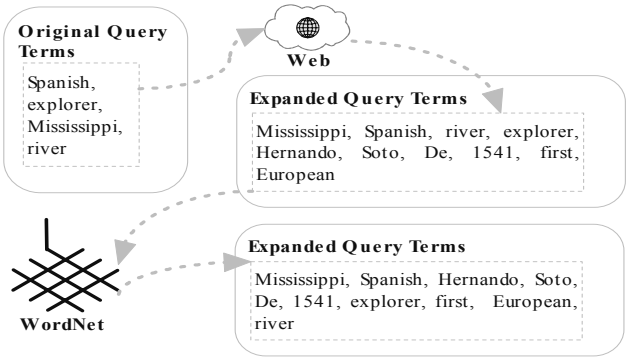


Figure 3: Example for Linear Query Formulation

4.3 Formulating Structured Queries

By examining the terms in $\underline{K}_q$, we found that most of them actually correspond to one or more QA event elements that we discussed in Section 2. In order to discover the structure of terms within the external knowledge, we perform structural analysis on $\underline{K}_q$ to form semantic groups of terms. We expect the semantic groups to match some forms of event structure in a QA event. Here, we employ a modified algorithm outlined in Liu & Chua [9] to perform the analysis.

Given any two distinct terms $t_i, t_j \in \underline{K}_q$, we compute their lexical, co-occurrence, and distance correlation as follows:

1) Lexical correlation: Here we consider only synonym relation, where:

$$R_l(t_i, t_j) = \begin{cases} 1, \text{ if } t_i \text{ and } t_j \in \text{ same synset;} \\ 0, \text{ otherwise.} \end{cases} \quad (4)$$

2) Co-occurrence correlation: Here we want to find the significant correlations between two terms, where:

$$R_{co}(t_i, t_j) = \frac{d_s(t_i \wedge t_j)}{d_s(t_i \vee t_j)} * \max\{0, p_{ij} - \frac{1}{\kappa_j}\} \quad (5)$$

where $p_{ij} = \frac{d_s(t_i \wedge t_j)}{\sum_{k=1}^{\kappa_j} d_s(t_k \wedge t_j)}$ (6)

where $d_s()$ functions are as defined in Eqn (1) and κ_j gives the number of terms in $\underline{K}_q$ that co-occur with t_j. Thus $1/\kappa_j$ denotes the average probability of term t_j co-occurring with other terms in $\underline{K}_q$ and p_{ij} denotes the normalized probability of t_i co-occurring with t_j. The $\max\{\}$ expression in Eqn (5) indicates that only those terms whose normalized co-occurrence probability is above $1/\kappa_j$ (or average) will have a positive co-occurrence value.

3) Distance correlation:

$$R_d(t_i, t_j) = \frac{1}{\frac{\sum |Pos(t_i) - Pos(t_j)|}{d_s(t_i \wedge t_j)}} \quad (7)$$

where $Pos(t_i)$ (or $Pos(t_j)$) denotes the position of term t_i (or t_j) in a web snippet or sentence. Thus $|Pos(t_i) - Pos(t_j)|$ gives the distance between two terms t_i and t_j in a web snippet or sentence. $d_s()$ function is as defined in Eqn (1). Hence, the distance correlation of two terms is denoted by the reciprocal of the average distance between them.

To explore the structure of external knowledge, we perform the following steps to derive the semantic groups of the terms in $\underline{K}_q$.

A) For each term $t_i \in \underline{K}_q$, derive a basic semantic group G_i which uses t_i as the main keyword. We denote t_i by *main*(G_i). Initially, we set $G_i=\varnothing$. By using the lexical, co-occurrence and distance correlations defined above, we derive the semantic relationship between t_i and any other term t_j in $\underline{K}_q$ as:

For each pair (t_i, t_j),

if $R_l(t_i, t_j)>0$, or, $R_{co}(t_i, t_j)>d_0$, or $R_d(t_i, t_j)>d_1$,
then $G_i \leftarrow G_i \cup \{t_j\}$.

where d_0 , d_1 are predefined thresholds.

At the end of the grouping process, we obtain a basic semantic group G_i. Note that $G_i \subseteq \underline{K}_q$.

B) Initialize the Basic Semantic Group Collection B , which contains the basic semantic groups for each term and Essential Semantic Group Collection E , which will contain semantic groups for all the terms:

$B \leftarrow \{G_1, G_2, ..., G_m\}$, $E \leftarrow \varnothing$.

C) Select a semantic group G_s in B, whose main word *main*(G_s) has the highest weight (See Eqn (1)) in $\underline{K}_q$:

$$i \leftarrow \arg\max_{s \; G_s \in B} (weight(main(G_s)))$$

D) Establish a dominant semantic group. Add G_s into E and eliminate G_s from B:

$B \leftarrow B - \{G_s\}$, $E \leftarrow E \cup \{G_s\}$.

E) Remove all terms found in G_s from remaining groups in B:

a) Eliminate those groups in B, whose main keyword is found in the selected group G_s. i.e.:

For each $G_i \in B$, if *main*(G_i) $\in G_s$, then $B \leftarrow B - \{G_i\}$.

b) Eliminate those terms in the remaining groups in B that are found in the selected group G_s. i.e.:

For each $G_i \in B$, $G_i \leftarrow G_i - G_s$.

F) If $B=\varnothing$, then output E as the final set of essential semantic groups; otherwise go to step C).

The set of resulting essential semantic groups will correspond loosely to the set of elements in a QA event. This assumption is reasonable as the Web and WordNet introduce world and lexical knowledge, which describe different aspects of an event thoroughly.

As a result of applying the above algorithm on the "*Mississippi*" example, we obtain the semantic groups as shown in the top left box of Figure 4. Some of the semantic groups show great cohesiveness among their terms such as "*Hernando, Soto, De"*. The terms within such groups are indispensable as representation to an element of a QA event. Other groups contain more loosely connected terms, which are actually different representations of the same element, e.g. *"French, Spanish"*.

To identify the semantic groups with the high cohesiveness, we compute the Group Cohesiveness *Coh*(G_i) as:

$$Coh(G_i) = \sum_{jk} \alpha_k R_k(main(G_i), t_j) \qquad (8)$$

where $\sum \alpha_k=1$, $R_k \in \{ R_l , R_{co} , R_d \}$, t_j is the j[th] term in group G_i, *main*(G_i) is the main keyword in G_i.

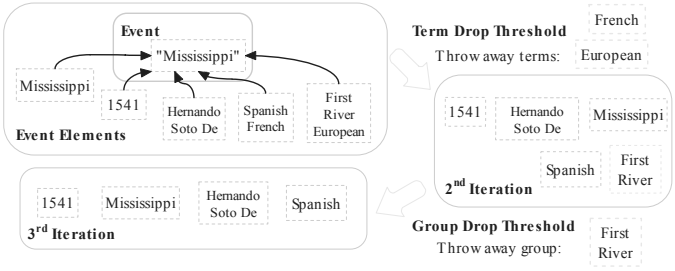


Figure 4: Example for Structured Query Formulation and Successive Constraint Relaxation

Basically, for terms within a highly cohesive semantic group, we treat them as a group of words that must co-occur simultaneously. Thus we use conjunction to connect them. For the rest of the groups that are less cohesive, we use disjunction to connect the terms. Finally, we use conjunction to connect between groups in order to maximize precision. The final Boolean query for the "*Mississippi*" example becomes: *"(Mississippi) & (French|Spanish) & (Hernando & Soto & De) & (1541) & (Explorer) & (first | European |River)"*.

One promising advantage of our approach is that we are able to answer any factual questions about the elements in this QA event other than just *"What Spanish explorer discovered the Mississippi River?"*. For instance, we can easily handle questions like *"When was Mississippi River discovered?"* and *"Which river were discovered by Hernando De Soto?"* etc with the same set of knowledge.

5. SUCCESSIVE CONSTRAINT RELAXATION (SCR)

Given the newly formulated linear or structured query, QUALIFIER uses the MG tool [17] to retrieve up to N (N=50) relevant documents within the QA corpus. It is well known that one of the disadvantages of Boolean retrieval is that it returns either too many or too few documents. For some cases, when the Boolean query is too strict, there is no good document returned and hence no answer will be found. However, this can be an advantage for QA as it ensures precision. In order to avoid over-constraining the query, we exploit a strategy called *Successive Constraint Relaxation* (SCR) to improve answer coverage while preserving precision. For the aforementioned *"Mississippi"* example, the resulting semantic groups for the first three iterations are shown in Figure 4. The algorithm works as follows:

For each question Q_i,

a) Get the original query vector $\underline{q}_i^{(0)}$.

b) Perform query formulation as described in Sections 4.2 & 4.3 to obtain the initial linear or structured expanded query vector $\underline{q}_i^{(1)}$.

c) Initialize the iteration counter L=1, and set L_{max} = 5.

d) Perform document retrieval and sentence ranking.

e) Perform answer extraction. If an exact answer of the correct type is found, then return the answer and exit.

f) Otherwise perform *successive constraint relaxation*. This is done by:

i) Computing a dynamic term selection threshold:

$$\tau_q^{(L)} = \frac{1}{1/L + \gamma * T_{mean} + \delta} * T_{\max} \qquad (9)$$

or a dynamic group selection threshold:

$$\varsigma_q^{(L)} = \frac{1}{1/L + \gamma * G_{mean} + \delta} * G_{\max} \quad (10)$$

where L is the number of iterations; γ and δ are constants depending on the question classes. T_{max}, T_{mean} are the max and mean weights of the terms in linear query $\underline{q}_i^{(L)}$ or in a semantic group of structured query $\underline{q}_i^{(L)}$; while G_{max}, G_{mean} are the max and mean average weights of the terms within the groups in structured query $\underline{q}_i^{(L)}$.

ii) Removing those terms (or groups) whose weights are below $\tau_q^{(L)}$ (or $\varsigma_q^{(L)}$).

For linear query, we have:
$\underline{q}_i^{(L+1)} = \underline{q}_i^{(L)} - \{\text{terms whose weights} <= \tau_q^{(L)}\}$

For structured query,
$\underline{q}_i^{(L+1)} = \underline{q}_i^{(L)} - \{\text{groups whose average weights} <= \varsigma_q^{(L)} \;\&\&\; \text{terms whose weights} <= \tau_q^{(L)}\}$

g) Increase iteration counter L by 1. If $L>=L_{max}$, then stop the process, otherwise go to step d).

h) Return "*NIL"* as the answer if there is no answer found at the end of the whole process.

As an alternative to Boolean retrieval with *successive constraint relaxation*, similarity-based search may be used to improve recall possibly at the expense of precision [19].

6. ANSWER EXTRACTION

With the set of the top K sentences obtained after document retrieval and sentence ranking, QUALIFIER performs fine-grained named entity tagging before extracting the string that matches the question class (or answer target or unknown element) as the answer.

QUALIFIER detects fine-grained named entities using a two-tiered hierarchy as shown in Table 2. It adopts a rule-based algorithm to detect the named entities by utilizing the lexical and semantic features such as capitalization, punctuation, context, part-of-speech tagging and phrase chunking. Our tests indicate that it could achieve a detection accuracy of over 90%.

Table 2: List of Fine-Grained Named Entities

HUMAN:	Basic, Organization, Person
TIME:	Basic, Day, Month, Year
LOCATION:	Basic, Body, City, Continent, Country, County, Island, Lake, Mountain, Ocean, Planet, Province, River, Town
NUMERIC:	Basic, Age, Area, Count, Degree, Distance, Frequency, Money, Percent, Period, Range, Size, Speed
OBJECT:	Basic, Animal, Breed, Color, Currency, Entertainment, Game, Language, Music, Plant, Profession, Religion, War, Works

Once an answer is found in the top i^{th} sentence, the system will stop the search for the rest of *(K-i)* sentences. For example, for question "*Where did Dr. King give his speech in Washington?*", we get:

- Q-class: ***LOC_BASIC***
- *<LOC_BASIC WASHINGTON>* KING-DREAM *<LOC_BASIC WASHINGTON>* In the *<NUM_ PERIOD 35 years>* since Dr. *<HUM_PERSON Martin Luther King>* Jr. delivered his "I Have a Dream" *speech* at the *<LOC_BASIC Lincoln Memorial>*, how have economic and social conditions changed for *<LOC_CONTINENT African>* Americans?

For question class ***LOC_BASIC***, we look for all the sub categories under *Location* and we pick ***Lincoln Memorial*** as the answer since it is the nearest (excluding zero distance) string to the original content word *speech* and is the most frequent *Location* NE in the top candidate sentences.

7. EVALUATION

7.1 TREC-11

The main task in TREC-11 [16] consists 500 factoid questions that seek exact answers from a QA corpus comprising over one million documents. Systems are required to return exactly one response per question, and order the question set by confidence in the response. A response is either a [answer-string; document-id] pair or the string "NIL". The "NIL" string will be judged correct if there is no answer known to exist in the corpus; otherwise it will be judged as incorrect. If a [answer-string; document-id] pair is given as the response, the answer-string must contain nothing other than the answer, and the document-id must be the id of a document in the collection that supports the answer-string as the answer. The pair will be judged as follows:

- correct: the answer-string consists of exactly a correct answer and that answer is supported by the document returned;
- unsupported: the answer-string contains a correct answer but the document returned does not support that answer;
- inexact: the answer-string contains a correct answer and the document supports that answer, but the string contains either partial answer or more than just the answer; and
- wrong: the answer-string does not contain a correct answer or the answer is not responsive.

Rather than the common precision measure, TREC-11 employed the *Confidence Weighted Score* (CWS) to reward systems that successfully rank questions that they answered correctly higher than those answered incorrectly. By using the linear query expansion as described in Section 4.2, QUALIFIER answers ***290*** questions correctly with a confident weighted score of ***0.610***. This places us among one of the top performing systems. Table 3 shows our system statistics obtained by the official NIST evaluation:

Table 3: Performance over 500 Questions in TREC-11

# correct	290	Precision	0.580
# unsupported	18	Confidence-weight score	0.610
# inexact	17	Precision of recognizing NIL	0.241
# wrong	175	Recall of recognizing NIL	0.891

7.2 Linear vs. Structured Query Formulation

To investigate the performance of using external resource like the Web and WordNet and the effectiveness of structured event-based QA approach, we conduct several tests to evaluate different uses of these resources as:

1) Baseline: We perform QA without using the external resources.
2) Baseline + Web: We add up to top m context words from $\underline{C}_q$ into $\underline{q}^{(1)}$ based on Eqn (3).
3) Baseline + Web + WordNet: We combine both Web and WordNet knowledge, but do not constrain the new terms

from WordNet. This is to test the effects of adding some WordNet terms out of context.

4) Baseline + Web + WordNet + context constraint: as defined in Section 4.2, i.e. linear query formulation.
5) Baseline + Web + WordNet + structure analysis: as defined in Section 4.3, i.e. structured query formulation.

For each test, we also compare the results obtained without performing Successive Constraint Relaxation (SCR) to that obtained after 5 iterations of SCR. In these tests, we examine the top 75 web snippets returned by Google with a cut-off threshold σ of 0.2. This set of parameters has been found in [19] to be most effective. We use the answer patterns and the evaluation script provided by NIST to score all the runs automatically. This may produce slightly different results from the NIST human assessment. For each run, we compute the precision, i.e. the number of correct answers over 500 TREC-11 questions. Table 4 summarizes the results of the tests.

Table 4: Precision of Different Query Formulation Methods

Test	Precision w/o SCR	Precision after 5 SCR Iterations
1	0.256	0.438
2	0.332	0.548
3	0.330	0.554
4	0.346	0.588
5	**0.472**	**0.634**

From Table 4, we can draw the following observations.

- Web-based query formulation (Test 2) improves the baseline performance by 25.1% in precision for the test with successive constraint relaxation. This confirms the results of many studies that using Web to extract highly correlated terms generally improves the QA performance.
- The use of lexical knowledge from WordNet without context constraint (Test 3) does not seem to be effective for QA, especially for the test w/o SCR. This is because it tends to add too many terms out of context.
- The linear query formulation that combines the Web and WordNet with context constraint (Test 4) achieves an accuracy of 0.588 after 5 SCR iterations. This gives an improvement of more than 7.3% over just using the Web (Test 2).
- The best performance is achieved (0.634) when carrying out the structured use of external knowledge (Test 5) as outlined in Section 4.3. It gives an improvement of 7.8% over the linear query formulation (Test 4) with SCR. The impact is much more significant for tests without SCR with a corresponding improvement of 36.4%.

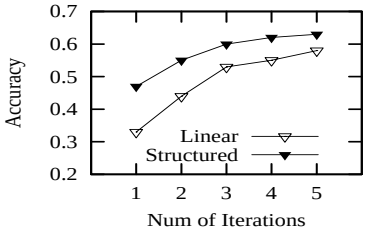


Figure 5: Results after every SCR Iteration

Figure 5 illustrates the effects of using *Successive Constraint Relaxation* iterations on the overall system performance for both linear and structured queries (i.e. Test 4 and Test 5). The results show steady improvements which indicate that our approach is effective.

7.3 Confidence Weighted Score

We studied the TREC-11 submitted runs, and found two interesting observations. First, the correctness of recognizing Non-NIL answers, i.e. answers which are not NIL, is much higher (around 30% to 50%) than the correctness of recognizing NIL answers in most of the high performance QA systems [5,6,11,15,18]. Hence the probability of NIL answers being recognized wrongly is higher than that of Non-NIL answers. Second, the abilities of a QA system to handle different types of questions are uneven. For our system, we found that for some question classes, like *Time, Location* and *Human*, QUALIFIER gives quite high performance (precision > 0.6); while for other question classes, like *Number, Description* and *Object*, the precision is low. (See Figure 6)

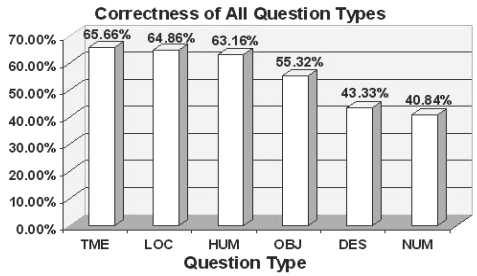


Figure 6: Correctness for Different Question Classes

Therefore, if we separate those NIL answers from Non-NIL answers and further rank both kinds of answers according to their question classes, we could improve the *CWS* of our system. We compute the *NIL-rate* for a question class C_i as follows:

$$\textit{NIL-rate}\ (C_i) = \frac{\#\,NIL-answers-in-C_i}{\#\,questions-in-C_i} \qquad (11)$$

NIL-rate actually measures a system's uncertainty about a question type. Therefore, answers to a low NIL-rate question class should be ranked higher than those to a high NIL-rate question class since the uncertainty of the low NIL-rate class is low and the confidence is high.

To achieve this, we first rank the confidence of answers to each question class based on its *NIL-rate* (C_i). We denote the resulting Class Confidence Rank as $\Re_{NIL\text{-}rate}(C_i)$. We then assign a priority level $\wp(Q_k)$ to each question Q_k based on its answer type (NIL or Non-NIL) and its question class C_i as follows:

$$\wp(Q_k) = \Re_{NIL\text{-}rate}(C_i) + A_{Type} * \aleph_c \qquad (12)$$

where A_{Type} is set to 1 if we have NIL as the answer, and 0 otherwise. $\aleph_c$ is number of question classes, which is 6 in our system if we consider only the general question types. Q_k belongs to question class C_i. Answers are then ordered based on the question's priority level. For those at the same priority level, we simply rank them according to the original question order.

For the "Martin Luther King" example, the $\Re_{NIL\text{-}rate}$ of its general Q-class *Location* is 2 and A_{Type}=0, hence the answer falls in the priority level of 2 (2=2+0*6).

Table 5 presents the value of *CWS* obtained for both the linear (our TREC submission) and structured queries using different

Table 5: Confidence Weighted Score by Different Ordering

Method \ Ordering	Unsorted	Upper Bound	Sorted
Linear	0.61	0.89	**0.80**
Structured	0.65	0.91	**0.82**

ways of ordering for the answers. The *Unsorted* case, which is used in our TREC submission, simply presents the answers in its original order without any modification. The *Upper Bound* gives the maximum CWS that we can achieve by manually putting all the correct answers before the wrong ones. The Sorted *CWS* uses the new ordering described earlier to re-rank the answers. As can be seen in Table 5, the sorting of answer is able to achieve a CWS value of **0.80** for the linear query formulation method. This is **23%** better than our TREC-11 submission. For the structured query formulation method, we could achieve a *Sorted* CWS of **0.82**, which represents the highest performance of our current system. The tests suggest that the re-ranking of answers based on prior knowledge of the confidence for different question classes can produce substantial improvement in CWS.

8. CONCLUSION

We have presented the QUALIFIER question answering system. QUALIFIER employs a novel approach to Event-based QA with the intuition that there exists implicit knowledge that connects an answer to a question, and this knowledge can be used to discover the information about a QA entity or different aspects of a QA event. Structural analysis is performed to extract semantic groups which correspond to different facets or elements of a QA event. Using the semantic groups to expand query, we are able to achieve an answer accuracy of 63.4%. We also improved our system's ability to rank answers based on the confidence and achieve a CWS value of 0.80 for the linear query formulation and 0.82 for the structured case. The results demonstrated that our approach is feasible and effective for open domain question answering.

9. REFERENCES

[1] AAAI Spring Symposium Series (2002). Mining Answers from Text and Knowledge Bases.

[2] ACL-EACL (2002). Workshop on Open-domain Question Answering.

[3] E. Brill, J. Lin M. Banko, S. Dumais, and A. Ng, "Data-intensive question answering", In Proceedings of the Tenth Text REtrieval Conference (TREC'2001), 393-400.

[4] C. Clarke, G. Cormack and T. Lynam, "Exploiting redundancy in question answering". In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2001), 358-365.

[5] C. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra and P. Tilker, "Statistical Selection of Exact Answers". In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 162-170.

[6] J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, D. Ferrucci "A Multi-Strategy and Multi-Source Approach to Question Answering", In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 124-133.

[7] E. Hovy, U. Hermjakob and C. Y. Lin "The use of external knowledge in factoid QA." In Proceedings of the Tenth Text REtrieval Conference (TREC'2001), 644-652.

[8] C. Y. Lin,"The Effectiveness of Dictionary and Web-Based Answer Reranking." In Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002).

[9] J. Liu and T. S. Chua, "Building semantic perceptron net for topic spotting", In Proceedings of 37th Meeting of Association of Computational Linguistics (ACL'2001), 370-377.

[10] D. Moldovan and V. Rus, "Logic Form Transformation of WordNet and its Applicability to Question Answering", In Proceedings of 37th Meeting of Association of Computational Linguistics (ACL'2001).

[11] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, O. Bolohan "LCC Tools for Question Answering", In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 144-154.

[12] M. A. Pasca and S. M. Harabagiu (2001). "High performance question/answering". In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2001), 366-374.

[13] J. Prager, E. Brown, A. Coden and D. Radev (2000). "Question answering by predictive annotation". In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2000), 184-191.

[14] D. R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal, "Probabilistic question answering from the web", In The Eleventh International World Wide Web Conference (WWW'2002).

[15] M. M. Soubbotin, S. M. Soubbotin, "Use of Patterns for Detection of Likely Answer Strings: A Systematic Approach", In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 134-143.

[16] E. M. Voorhees. "Overview of the TREC 2002 Question Answering Track." In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 115-123.

[17] I. Witten, A. Moffat, and T. Bell, "Managing Gigabytes", Morgan Kaufmann, 1999.

[18] H. Yang and T. S. Chua, "The Integration of Lexical Knowledge and External Resources for Question Answering", In notebook of the Eleventh Text REtrieval Conference (TREC'2002), 155-161.

[19] H. Yang and T. S. Chua, "QUALIFIER: Question Answering by Lexical Fabric and External Resources", In the Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'2003), 363-370.