# QUALIFIER in TREC-12 QA Main Task

**Hui Yang, Hang Cui, Mstislav Maslennikov, Long Qiu, Min-Yen Kan, Tat-Seng Chua**

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
Email: {yangh, cuihang, maslenni, qiul, kanmy, chuats}@comp.nus.edu.sg

### Abstract

This paper describes a question answering system and its various modules to solve definition, factoid and list questions defined in the TREC12 Main task. In particular, we tackle the factoid QA task by Event-based Question Answering. Each QA event comprises of elements describing different facets like *time, location, object, action* etc. By analyzing the external knowledge from pre-retrieved TREC documents, Web documents, WordNet and Ontology to discover the QA event structure, we explore the inherent associations among QA elements and then obtain the answers. There are three subsystems working parallel to handle definition, factoid, and list questions separately. We highlight the shared modules, fine-grained named entity recognition, anaphora resolution and canonicalization co-reference resolution, among the three subsystems as well.

## 1 Introduction

Open domain Question Answering (QA) is a complex research area formed as a distinctive combination of Information Retrieval (IR), Information Extraction (IE), and Natural Language Processing (NLP). The basic problem that QA poses is: given a question and a large text corpus, return an "answer" rather than relevant "documents". QA has received tremendous interest recently with the emergence of many commercial products, and research papers in various communities (SIGIR 2003, ACL 2003, ACMMM 2003). In TREC-11 (Voorhees 2002), 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 (Yang&Chua,2002). 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 to further improve the performance.

This year, we model the factoid questions by Event-based Question Answering (Yang et al. 2003). Questions often refer to several aspects/elements of the events, such as Location, Time, Subject, Object, Quantity, Description and Action, etc. For most QA events, there are inherent associations among their elements. We thus perform *Event Mining* to discover and then incorporate the knowledge of event structure systematically for more effective QA.

Our system, named **QUALIFIER (QUestion Answering by LexIcal FabrIc and External Resources)**, includes modules to perform detailed question analysis, QA event construction, answer justification, fine-grained named entity recognition, anaphora resolution, canonicalization co-reference, and successive constraint relaxation.

During question parsing, detailed question classes, answer types, original query terms and NLP roles of the query terms are analyzed. We derive detailed question class ontology that corresponds to fine-grained named entities. This enables us to extract exact answer from the candidate sentences more accurately. All the questions are treated equally during the stage of detailed question analysis, and then passed to three different subsystems to handle definition, factoid and list questions separately. The original query terms can be used directly to locate potential answer candidates in the corpus. However, one major problem is that those terms do not provide sufficient evidence to retrieve the answer candidates. This is known as the semantic gap between the query space and document space. In order to bridge this gap, we use the knowledge of both the Web and lexical resources to expand the original query. The new query therefore contains terms that are related to the local context in the Web and the lexical context in WordNet. Finally, we structure the query and use it to search for answer candidates through the MG system (Witten et al. 1999). Answer candidate sentences are selected from the top returned documents and are ranked based on association rules obtained from QA Event analysis. Named entity recognition, answer justification, canonicalization resolution and answer selection are done to extract the final answer while successive constraint relaxation is used as an auto-feedback loop to boost the answer coverage.

We will describe the three subsystems one by one in the following sections. Figure 1 illustrates the flow of the main system. For factoid and list questions, they are solved similarly and definition questions are handled differently in answer extraction.

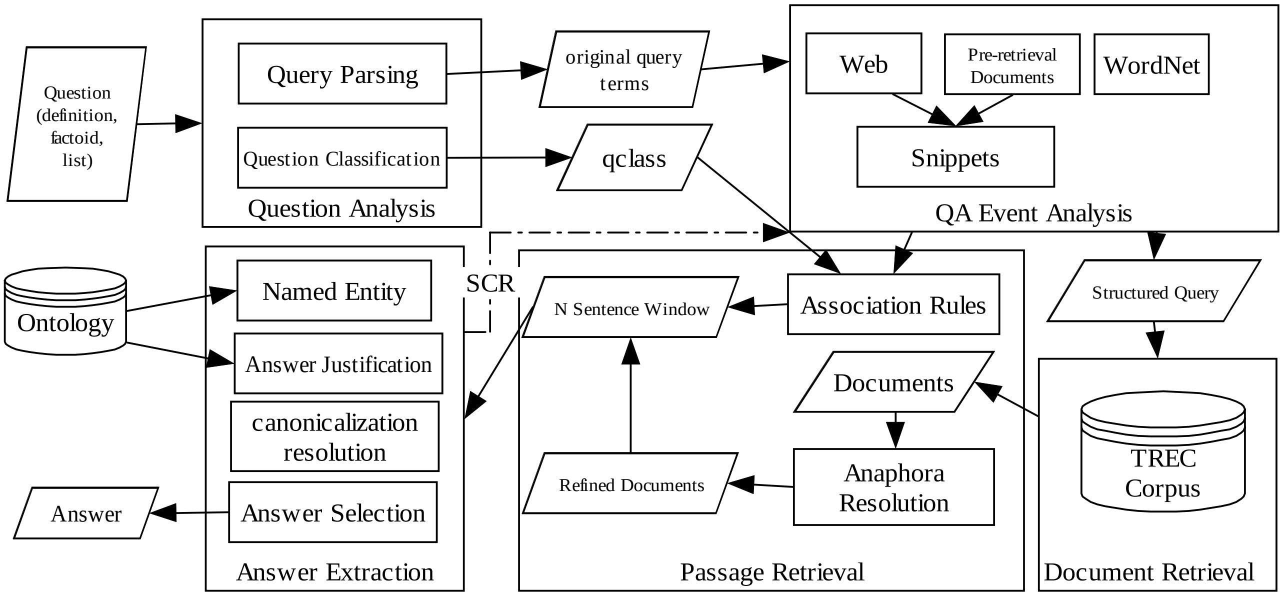

Figure 1: Overview of QUALIFIER

## 2 Factoid Questions

Our system performs event-based question answering for factoid questions in a few steps: external knowledge acquisition, QA event construction, query formulation, element association mining and answer processing. First, it 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). During the knowledge acquisition stage, it integrates the knowledge of the pre-retrieved TREC documents, Web, WordNet, and our manually constructed Ontology to extract additional evidences for the query. Second, it performs event construction to discover different facets or elements of events and employs the knowledge of events to perform query formulation. Third, given the newly formulated query, it employs the MG tool to search for top ranked documents in the corpus. Fourth, for the top ranked documents, it identifies the relevant passages by exploiting the associations among event elements. After performing element association mining, it computes the Answer Event Score (AES) and uses it to rank the passages from the relevant documents in the corpus. Answer justification module reinforces the confidence of the returned correct answers and filters out some unreasonable ones.

### 2.1 QA Event Mining

In our previous work, we modeled the world and lexical knowledge from the Web and WordNet to support effective QA. Basically, we performed the structured analysis of the external knowledge to extract the QA event structure. First, we used the original query terms in the questions to retrieve the top $N_w$ documents by using the Web search engine and then extracted the terms that are highly correlated to the original query terms. Second, we used WordNet to adjust the term weights as well as to introduce new lexical related terms. Third, we computed the lexical, co-occurrence, and distance correlations between terms and used these as the basis to induce event elements by unsupervised semantic grouping. For example, given the question, “*What Spanish explorer discovered the Mississippi River?*”, we could get the event structure as shown in Figure 2. Finally, we used this event structure to formulate boolean query to retrieve relevant documents from the QA corpus, and employed a featured-based approach to perform answer selection and extraction.

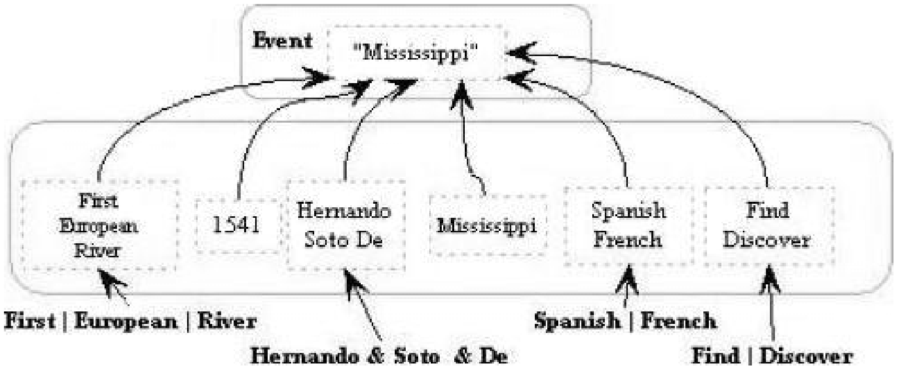

Figure 2: Example for QA Event Structure

After extracting the event structure, we employ *Event Mining* to study the relationships among the event elements. In this approach, we extract important association rules among the elements by using data mining techniques. Given a QA event $E_i$, we define X, Y as two sets of event elements. Event mining studies the rules of the form $X \rightarrow Y$, where "→" means "implies", and Y is the possible answer candidate set. In order to avoid too many misleading rules, we restrict the relationships before generating the rules:

if $X \subseteq Y$ , ignore $X \rightarrow Y$.
if cardinality(Y) > 1, ignore $X \rightarrow Y$.
if $Y \cap \{\text{element}_{\text{original}}\} \neq \emptyset$, ignore $X \rightarrow Y$.

Also, we define *Event Mining* as follows:

***Event mining*** studies the association rules of the form $X \rightarrow Y$, where X, Y are QA event element sets, $X \cap Y = \emptyset$, and $Y \cap \{\text{element}_{\text{original}}\} = \emptyset$.

There are 4 major steps in the event-based QA approach and we are going to elaborate the details in the following subsections.

a) Extract the QA event from the Web and WordNet for a question.
b) Mine the event in a) to generate the useful association rules among the event elements.
c) Rank the passages in the relevant documents from the QA corpus by matching them with the association rules.
d) Extract the answer phrase from the top passages.

## 2.2 QA Event Generation

We explore the use of semantic grouping to structurally utilize the external knowledge extracted from the Web, WordNet, Ontology and pre-retrieved documents from TREC. The terms extracted from the relevant web snippets, WordNet, pre-retrieved TREC documents are put into a vector called $\underline{K}_q$, which is the basis for semantic grouping since it is most likely to contain the facts/terms about the QA entity or QA event. Given any two distinct terms $t_i$, $t_j$, in the $\underline{K}_q$ , we compute (a) Lexical correlation $R_l(t_i, t_i)$;(b) Co-occurrence correlation $R_{co}(t_i, t_j)$; (c) Distance correlation $R_d(t_i, t_j)$. With the term association measures $R_l$, $R_{co}$, and $R_d$, we employ an unsupervised clustering algorithm (yang et al. 2003).to derive the semantic groups of the terms in $\underline{K}_q$, which are expected to match with the event structure. Figure 2 shows the QA event structure developed after performing knowledge modeling.

## 2.3 Association Rule Mining and Answer Extraction

After constructing the events from the textual data, event-mining techniques are applied to discover association rules within the QA events. We perform the rule mining in two steps: association rule generation and selection. Association rules are in the form of $X \rightarrow Y$ as we mentioned earlier. Basically, we need to find all the combinations of the event elements to form the sets X and Y. To do this, we need to assign Y to contain only one of the QA event elements. For the rest of the elements, we use them to form different X in various sizes and combinations. We then store all of the $X \rightarrow Y$ rules in an association rule bank $B_i$ for QA event $E_i$. The association rule generation algorithm is given as follows.

```
Algorithm Association_Rule_Mining ($E_i$)
Input: QA event $E_i$ and event element set $S_i$
Output: Association rule bank $B_i$ of QA event $E_i$
for each element $e_j$ in $S_i$
  $Y = \{ e_j \}$
  $G_j = S_i - Y$
  for (k=1; k=cardinality($G_j$); k++)
    select k elements from $G_j$ to form a
    collection of size-k element sets $R^{(k)}$;
    for each size-k element set $r_m^{(k)}$ in $R^{(k)}$
      $X = r_m^{(k)}$
      add $X \rightarrow Y$ into rule bank $B_i$
Return $B_i = \{X \rightarrow Y | X \in \cup_k R^{(k)}\}$;
```

Usually for a QA event with n elements, the number of association rules $\aleph_{\text{rules}}$ is:

$$\aleph_{\text{rules}} = n * \sum_{k=1}^{n-1} C_k^{n-1} \qquad (1)$$

The number of these rules is potentially large and hence it is necessary to prune away some rules that do not have “interesting” information. Note that not all the rules are reliable. This is because:

- Some association rules are not complete. For example, above algorithm will discover both rule X → Y and rule Z → Y, where $X \subseteq Z$. Then X → Y is not as complete as Z → Y. Thus we have to decide which one should be preserved to extract more accurate answers.
- The rules appear fewer times may not be so significant for a certain QA event. The more often a rule appears in the document collection, the more important it might be.

In order to identify the interesting association rules among the event elements, we need to measure the “usefulness” of the rules. The rules containing more information and involving more event elements are considered to be more important. We consider the typical *Support* measure in data mining (KDD) literature (Dörre et al. 1999) and the reliability of the event elements in X. Therefore, the first measure *Support (X → Y)* is given as:

$$\frac{d_w(X \wedge Y)}{\aleph_{window}} + \alpha * \aleph_{original}(X) + \beta * \aleph_{\exp anded}(X) \qquad (2)$$

where $\alpha + \beta = 1$, $\aleph_{original}(X)$ is the number of elements containing the original question terms in X, $\aleph_{\exp anded}(X)$ is the number of elements containing the expanded query terms in X, $d_w(X \wedge Y)$ is the number of passages containing both X and Y, and $\aleph_{window}$ is the total number of passages in the documents.

Another measure is *Confidence,* which helps to filter out both false information introduced by unreliable data source and the conflicting rules. *Confidence (X → Y)* as defined in data mining:

$$\frac{d_w(X \wedge Y)}{d_w(X)} \qquad (3)$$

We select the best association rules for each event element based on these two measures, which indicates the “usefulness” of the event element relationships. These association rules are combined with event element matching to rank the passages. We compute the *Answer Event Score* (AES) for each passage from the relevant documents in the QA corpus. It is defined as:

$$\frac{M_{ele} + \sum_{i=1}^{N_r} (M_i * (Support(rule_i) + Confidence(rule_i)))}{N_{ele}} \qquad (4)$$

where $M_{ele}$ is the number of matched event elements; $Support(rule_i)$ ( $Confidence(rule_i)$ ) is the support (confidence) for the matched rule i; $M_i$ is set to 1 when rule i is present in this passage, otherwise it is set to 0; $N_r$ is the total number of association rules, and $N_{ele}$ is the total number of event elements for the question.

The good passages we have now describe the same event as the question. Hence, once we have these passages, we can either use the event element in Y as the answer or extract the answer from the passages. In order to extract the exact answers from the passages, we first apply named entity tagging on the top ranked passages and the event association rules. All the named entities whose type match with answer target are selected as answer candidates and ranked based on AES score of the passage where they extracted from, number and significance of the association rules that they match. The ranking formula for answers candidate j is defined as:

$$AES(P_j) + \sum_{i=1}^{Y_j} Support(rule_i) \qquad (5)$$

Where $Y_j$ is the number of top association rules whose Y is matched with *j;* $Support(rule_i)$ is the support for the rule i; AES(Pj) is the AES score of the passage where answer candidate j extracted from.

## 2.4 Fine-grained Named Entity Recognition

With the set of the top passages 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. The system 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.

The input text is going through the preprocessing stage. We split sentences and remove all characters, which potentially cannot be seen or may risk the following process (mainly, those are non-ASCII characters). In addition, we extract some simple types on this stage, like NUM_PERCENT or COD_URL. In order to avoid the problems with calculations, the program may transfer the text presentation of numbers to numerical. For example, it will convert “one hundred eleven” to 111. The output is represented in the XML format, which contains some marked named entities. We then use the shallow parser by the company Infogistics. The program performs part-of-speech tagging and extracts noun groups and verb groups.

The named entity extraction module is the core for the whole system. The heuristic rules allow creating user-defined types. Also, they support the regular expression style for features of words. Example of the possible rule:

person_title_np = listi_personWord src_, hum_Cap2+ src_, $set(HUM_PERSON/2)

This rule sets the type for each noun phrase, which contains some known person title (academic, academician etc.). The assigned type is person_title_np for this particular phrase. $set(HUM_PERSON/2) means, that the second position (positions are started from 0) should be settled to HUM_PERSON.

Interpretation of those rules starts from the lists initialization. Each document is parsed sentence by sentence. During the parsing stage, each word is represented in the form of token with several features. The output of the interpreter may involve several competing types for some named entity. We also use the following heuristics to choose the rule in such situation:

1) Longer length. We preferred to extracted longer named entities
2) Ontology. If the rule is found in some resource (e.g., confirmed list of persons), then we assigned this type of rule;
3) Handcrafted priorities, which disambiguates the named entities correctly in most of the cases.

Our NE recognizer supports 51 types of NEs, which support 2 levels of granularity. Top levels of classification are human (HUM), location (LOC), number (NUM), object (OBJ), time (TME) and code (COD). The module extracts named entities based on the set of heuristic rules and lists for the semantic categories. We added some new types of fine-grained NE into our old system. QUALIFIER detects fine-grained named entities using a two-tiered hierarchy as shown in Table 1.

**Table 1: Partial 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 |
| **NUMBER**: | Basic, Age, Area, Count, Degree, Distance, Frequency, Money, Percent, Period, Range, Size, Speed |
| **CODE** | URL, Telephone, Post code, email address, Product index |
| **OBJECT**: | Basic, Animal, Breed, Color, Currency, Entertainment, Game, Language, Music, Plant, Profession, Religion, War, Works |

In last year’s TREC evaluation for NUM questions, we got the NE tagging for NUMBER only 17 correct out of 29 instances, or an accuracy of 58.6%, which is the lowest rate among all the question types. In order to improve QUALIFIER’s ability to recognize NUMBER, we added a range converter module into NE recognizer. We unify numbers close to each other into ranges other than the absolute surface numeric value appearing in the corpus. E.g. “5000000” should be the same range as “5.1 million”.

## 2.5 Anaphora Resolution

In order to maximize recall of the system, which is crucial for list questions, we perform anaphora resolution for the retrieved document by MG system before we proceed to passage selection.

Firstly, we make use of Charniak’s parser (Charniak, 2000) to generate the parse tree for each sentence. The Parse Tree Walker extracts two lists. One list contains all the NPs in the text and the other contains all the anaphors. Their agreement features, head-argument/head-adjunct information and all the salience factors except sentence recency are annotated. All the pleonastic pronouns are filtered out. Their antecedent is assigned as null.

For each lexical anaphor, it forms a pair with each item in a subset of the NPs (currently the implementation only considers NPs contained within three sentences proceeding the anaphor and those in the sentence where the anaphor resides). These antecedent candidate-anaphor pairs are examined by the Anaphora Binding Algorithm. For third person pronouns, similarly, the syntactic filter is applied on the candidate pairs consisting one pronoun and one NP from the set satisfying the same positional criteria. In both cases a morphological filter is applied, checking the agreement features compatibility.

For the candidates identified earlier, their salience weights are computed and they are ranked by an arbitrator accordingly. The candidate with the highest weight is proposed as the actual antecedent. In case of tie, the one closer to the anaphor is favored. Figure 3 shows the structure of the anaphora resolution process.

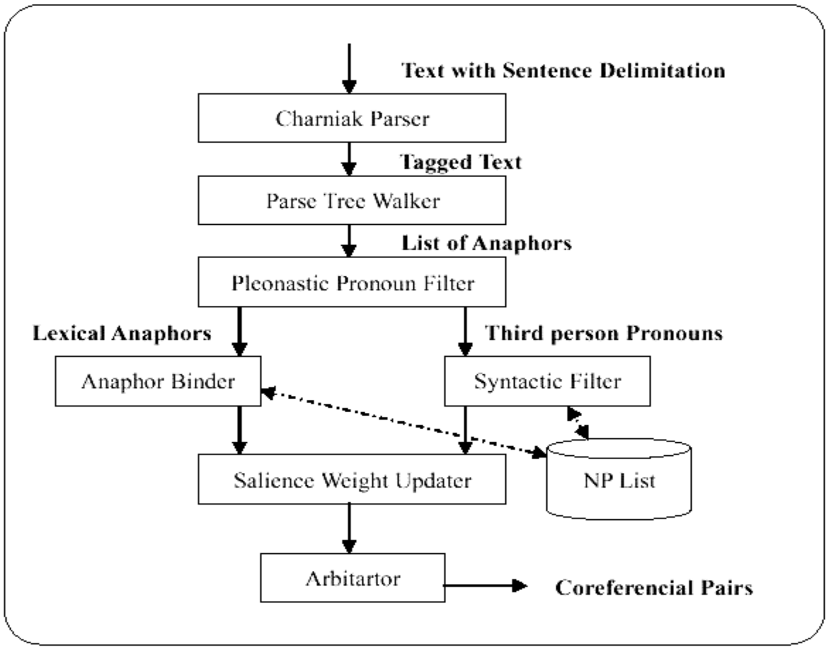

Figure 3: Anaphora Resolution

### 2.6 Answer Justification

Answer Justification attempts to establish lexica-semantic paths between the questions and candidate long answers, and derive logical proofs along those paths to justify whether the answer is correct. The prover uses two types of axioms. The first type is used to guide the proof and includes axioms derived from facts in the textual answers. The other type of axioms generated based on our manually constructed ontology. For example,

q1425: What is the population of Maryland?
Sentence: “Maryland 's population is 50,000 and growing rapidly.”
**Ontology Axiom (OA):** Maryland (c1) & population (c1, c2) -> 5000000(c2)

In this way, we could identify the wrong answer “50000”, which is what the surface text shown. Only when we know certain constraint to indicate the actual range of Maryland’s population, we know that the surface answer is wrong.

## 3 List Questions

List question answering is quite similar to what we did for factoid questions. However, we allow the answer extraction module to return multiple answers and remove duplicated answer candidates with the help of abbreviation co-reference. The exact answers are exacted based on patterns and its ranking. Each of these answers is passed to answer justification module for confirmation.

List questions are processed similarly to factoid questions. However, we allow the answer extraction module to return multiple answers and remove duplicated answer candidates with the help of abbreviation co-reference. In general, our method is more data-driven because we want it to be domain independent. However, we still utilized some heuristics to enhance the process.

From a list question, we obtain the same or less amount information comparing to factoid questions since the constraint from the list question itself will be more relax. Under this situation, possible answers for a list question could be in any number. It’s hard for us to decide when the right time to stop is. The only way to abase this uncertainty is to perform exhaustive search or near exhaustive search. These patterns could mean surface text patterns, commonly used cue words. There are some examples:

- <same_type_NE>, <same_type_NE> and <same_type_NE> + verb …
- … include: <same_type_NE>, <same_type_NE>, <same_type_NE> …
- “list of …”
- “top” + number + adj-superlative
- “alphabetical list” of …
- “following list…”

## 4 Definition Questions

Due to the characteristics of definitional questions, i.e. informative and stress on completeness, we treat answering definitional questions as an integrated process of information retrieval and summarization. We utilize techniques from information retrieval as anti-noise mechanism and make use of summarization techniques to avoid redundancy in the results. The pipeline system can be divided into 2 modules: *sentence ranking, and sentence selection plus summary generation.* Figure 4 shows an overview of the definition question subsystem.

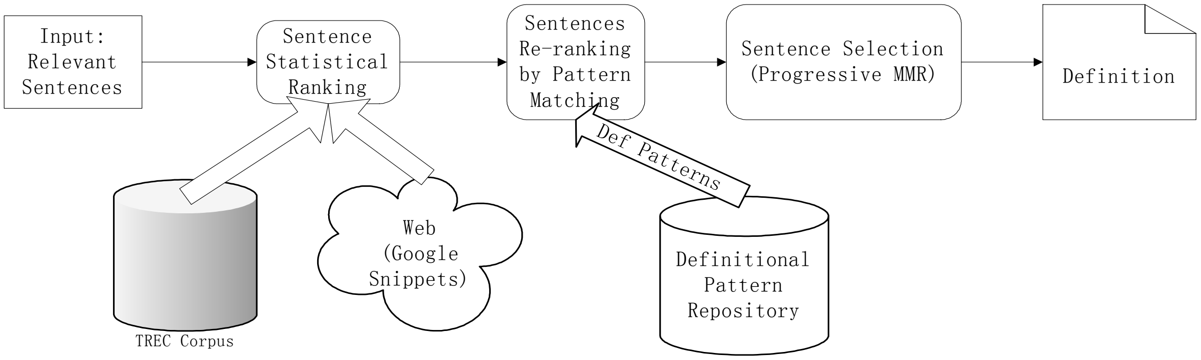
Input: Relevant Sentences
Sentence Statistical Ranking
Sentences Re-ranking by Pattern Matching
Sentence Selection (Progressive MMR)
Definition
Def Patterns
Web (Google Snippets)
Definitional Pattern Repository
TREC Corpus

Figure 4: Illustration of Definition Subsystem

Due to the heterogeneity of the news articles, many of the input documents are actually not related to the search target. We applied a document filter to the get only those documents containing all terms of the search target are labeled as “relevant”. We then applied anaphora resolution to sentences from all “relevant” documents to replace appropriate pronouns with the search target. Finally, those sentences containing any part of the search target and their contextual sentences (one sentence proceeding and following them respectively) are sifted as “positive set”. All other sentences in the input documents are considered as “negative set”.

As for sentence ranking, we utilized evidence from two sources, namely the input documents and the Web. Basically, we use sentence frequency (the number of sentences containing the word) as the main metric to measure the importance of each word. The sentences in “negative set” are used to provide “negative examples”. In other words, a word is considered important if it appears often in the sentences of “positive set” while occurs rarely in the “negative set”. This can be done in a TFIDF-like fashion:

$$Weight_{Corpus}(s) = \log(1+\sum_{w\in s} CorpusSF_{Positive}(w)\times \log(1+\frac{\#\,Negative\,Sentences}{CorpusSF_{Negative}(w)})) \qquad (6)$$

In order to account for the diversity of the news articles, we use the Web as a supplementary source. The Web evidence is derived from the snippets obtained using the queries constructed from the sentences of the “positive set” containing any part of the search target. Specifically, the expansion terms are selected by:

$$Weight_{Exp}(w) = \frac{SF(w)}{\#Total\ Sentences}\times \log(1+\frac{Co(sch\_term, w)}{f(w)+f(sch\_term)}\quad) \qquad (7)$$

The weight of the sentence for Web evidence can be expressed as:

$$Weight_{Web}(s) = \sum_{w}\log(1+Web\_SF(w))\times \log(1+\frac{\#\,PositiveSentences}{Corpus\_SF_{Positive}(w)}) \qquad (8)$$

These two weight values are linearly combined to represent the sentence weight for the search target.

$$Weight(s) = \lambda\cdot Weight_{Corpus} + (1-\lambda)\cdot Weight_{Web} \qquad (9)$$

After sentence ranking, we have a list of ordered sentences with the most relevant sentences to the search target being placed in the top of the list. We made use of summarization techniques to accomplish sentence selection because sentences from news articles are likely to contain duplicated content. We employed a variation of the Maximal Marginal Relevance (MMR) to select sentences from the list while avoiding redundancy between the summary sentences:

a) All sentences are ordered in descending order by weights.
b) Add the first sentence to the summary.
c) Examine the following sentences.
If *Weight(stc)- Weight(next_stc)* < $\beta$ *avg_sim(stc)* continue;

else Add stc to summary, where $avg\_sim(stc) = \frac{\sum_{s \in Sum} Sim(s, stc)}{\# sum}$ (10)

*avg_sim(stc)* is the average similarity of the sentence *stc* and the sentences already in the summary. The similarity is defined as the measure of their word overlapping as:

$$Sim(s, stc) = \sum_{w \in s \cap stc} \log(1 + \frac{\# All\_matching\_stc}{2 \times TREC\_SF_{all\_matching}(w)}) \tag{11}$$

d) Go to Step c) till the length limit of the target summary is satisfied.

In general, our method is more data-driven because we aim it to be domain independent. However, we still utilized some heuristics to enhance the process. For example, if a sentence containing "<*Target*>, a ……" or "<*Target*>, which is the ……", it is likely to be a good definitional sentence for the target. Thus in the sentence ranking and content selection processes, we employed a set of heuristic rules. In content selection process, we applied heuristics to selecting meaningful fragments of the summary sentences to construct the final answers for the search target.

## 5 Results

We submitted 3 runs for TREC 2003 QA main task. They are mmlnus03r1 (definition run 1 + factoid run 1 + list run 1), mmlnus03r2 (definition run 2 + factoid run 1 + list run 1) and mmlnus03r3 (definition run 3 + factoid run 2 + list run 1). The first run mmlnus03r1 emphasized more on recall (answer coverage) while the third run mmlnus03r3 more on precision (answer accuracy). The second run was a hybrid to test the balance between recall and precision.

The 2 factoid runs focus on precision and recall each. Our first run maximizes recall (answer coverage) by using anaphora resolution, abbreviation co-reference, and successive constraint relaxation to find as many answers as possible without answer justification. In the second run, however, we focused on precision (answer correctness), answer justification plays a rather important role and successive constraint relaxation keeps the recall at a satisfactory level. Table 2 gives the scores for our factoid runs.

The 3 definition runs that we submitted to TREC balance precision and recall. We empirically set the length of the summary for people and objects. Our first run maximizes recall by using full sentences. We have the same length limit for summary in run 2, while text fragments are extracted by heuristics instead of using the full sentences. In the third run (to maximize precision), fewer sentences were extracted for summary and text fragments extraction was also applied. The performance of our definition runs is given in Table 3.

Only one run of list questions is submitted. Its average precision over 37 questions is given in Table 4.

Table 5 summarizes the overall performance for the three submitted runs. It shows that nulmmlr2 gives the best performance, which uses strict constraints and focuses on answer accuracy.

**Table 2: Performance over 413 Factoid Questions in TREC-12**

| | | | | |
|---|---|---|---|---|
| nusmmlr1<br>nusmmlr2 | # correct | 232 | Accuracy | 0.562 |
| | # unsupported | 24 | Precision of recognizing NIL | 0.160 |
| | # inexact | 13 | Recall of recognizing NIL | 0.400 |
| | # wrong | 144 | | |
| nusmmlr3 | # correct | 225 | Accuracy | 0.545 |
| | # unsupported | 20 | Precision of recognizing NIL | 0.158 |
| | # inexact | 12 | Recall of recognizing NIL | 0.767 |
| | # wrong | 156 | | |

**Table 3: Performance over 50 Definition Questions in TREC-12**

| | | |
|---|---|---|
| Average F score for definition question | nusmmlr1 | 0.441 |
| | nusmmlr2 | 0.473 |
| | nusmmlr3 | 0.432 |

**Table 4: Performance over 37 List Questions in TREC-12**

| | | |
|---|---|---|
| nusmmlr1<br>nusmmlr2<br>nusmmlr3 | Average precision | 0.568 |
| | Average recall | 0.264 |
| | Average F1 | 0.317 |

**Table 5: Overall Performance of 3 Submitted Runs in TREC-12**

| | |
|---|---|
| nusmmlr1 | 0.471 |
| nusmmlr2 | 0.479 |
| nusmmlr3 | 0.460 |

## 6 Conclusion

We develop three subsystems for this year’s TREC. Definition questions answering combines techniques from information retrieval as anti-noise mechanism and make use of summarization techniques to avoid redundancy in the results. Factoid question and list questions take the advantage of Event-based QA, which employs the intuition that there exists implicit knowledge that connects an answer to a question, to extract the correct answer. Association rules are mined to get the relationship among the QA event elements. The whole system consists many modules, including detailed question analysis, QA event construction, event mining, document retrieval, passage retrieval, answer extraction, answer justification, fine-grained named entity recognition, anaphora resolution, canonicalization co-reference, and successive constraint relaxation. We also manually constructed Ontology to lever the performance of our NE and answer justification modules.

We are currently refining our approach in several directions. First, we are trying to formulate a formal proof of our QA event hypothesis. Second, we are working towards an online question answering system. Our longer-term research plan includes Interactive QA, and the handling of more difficult analysis and opinion question types.

## Acknowledgement

We would like to thank Dr. Jimin Liu, Ms Jing Xiao and Mr. Chun-Keat Koh for their great efforts in the earlier development for the named entity recognition module. Also, we would like to thank our student assistants Yee-Fan Tan, Victor Goh and Shi-Yong Neo for constructing Ontology and conducting various experiments. This work was supported in part by the A*Star and NUS Joint-Lab funding.

## References

E. Charniak. 2000. “A maximum-entropy-inspired parser”, In Proceedings of the first 1st Conference of the North American Chapter of the Association for Computational Linguistics, (NAACL’2000).

Jochen Dörre, Peter Gerstl, Roland Seiffert.1999.”Text mining: finding nuggets in mountains of textual data”, In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD’1999), 398-401.

I. Witten, A. Moffat, and T. Bell, 1999, “Managing Gigabytes”, Morgan Kaufmann, 1999.

Infogistics, http://www.infogistics.com/posdemo.htm

E.M.Voorhees. 2002. “Overview of the TREC 2002 Question Answering Track.” In the Proceedings of the Eleventh Text REtrieval Conference (TREC’2002), 115-123.

H. Yang , T. S. Chua, 2002. “The Integration of Lexical Knowledge and External Resources for Question Answering”, In the Proceedings of the Eleventh Text REtrieval Conference (TREC’2002).

H. Yang , T. S. Chua, S Wang, C Koh, 2003. “Structured Use of External Knowledge for Event-based Open Domain Question Answering”, In the Proceedings of the Twenty-Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR’2003).