# Experiments with Web QA System and TREC2004 Questions

**Dmitri Roussinov**
Arizona State University
Department of Information Systems
Dmitri.Roussinov@asu.edu

**José Antonio Robles-Flores**
Arizona State University
Department of Information Systems
Jose.Robles@asu.edu

**Yin Ding**
Computer Science Department
Arizona State University
Tempe, Arizona 85287
yding@asu.edu

## ABSTRACT

We describe our first participation in TREC. We only competed in the Question Answering (QA) category and limited our runs to factoids. Our approach was to use our open domain QA system that finds the answer among Web pages indexed by a commercial search engine, then project the answer to the TREC test collection. Our Web QA takes advantage of the redundancy of the Web, obtains the candidate answers by pattern matching, then performs probabilistic triangulation of them to assign a final score. Our novel contributions are the following 1) the probabilistic triangulation algorithm, 2) a more powerful pattern language than used in prior research, and 3) use of semantic features of expected answers instead of relying on an elaborate hierarchy of question types. Although, we were able to run only first 91 out of 230 factoid questions before the submission deadline, we find our result encouraging, and if interpolated to the entire questions set, it would have placed us above the median performance on factoid questions.

## INTRODUCTION

Our participation in TREC was essentially a by-product of our involvement in a completely Web based open domain question answering project (Roussinov & Robles, 2004). We maintain a working demo on the Web (http://129.219.59.31/qademo.html), which has averaged 15 requests a day since the inception in August 2004. Our QA system is entirely web based: it finds the answers among the web pages available through the general purpose search engine indexes (GPSE) such as Google, AltaVista, AOL, MSN etc., without even using the TREC collections. For the TREC QA competition, it then attempts to find the best supporting document within the TREC collection. We extended the approach by (Dumais et al., 2002) by automatically learning patterns similar to suggested by the other researchers (Ravichandran & Hovy, 2002; Zhang & Lee, 2002), and by using a more powerful pattern language. Another distinction from many systems presented earlier at TREC, is that we do not use an extensive hierarchy of question types organized by the expected semantic category of an answer (e.g. *person, celebrity, organization, city, state*, etc.), but rather we use a small set of independently considered semantic features, presence of which in candidate answers only affects their score rather than excludes entirely.

## ALGORITHMS AND IMPLEMENTATIONS

Our overall Web based QA system is shown on the Figure 1. For pattern training and answer matching it uses an underlying general purpose search engine (GPSE). We currently use Google, AltaVista, and AOL.

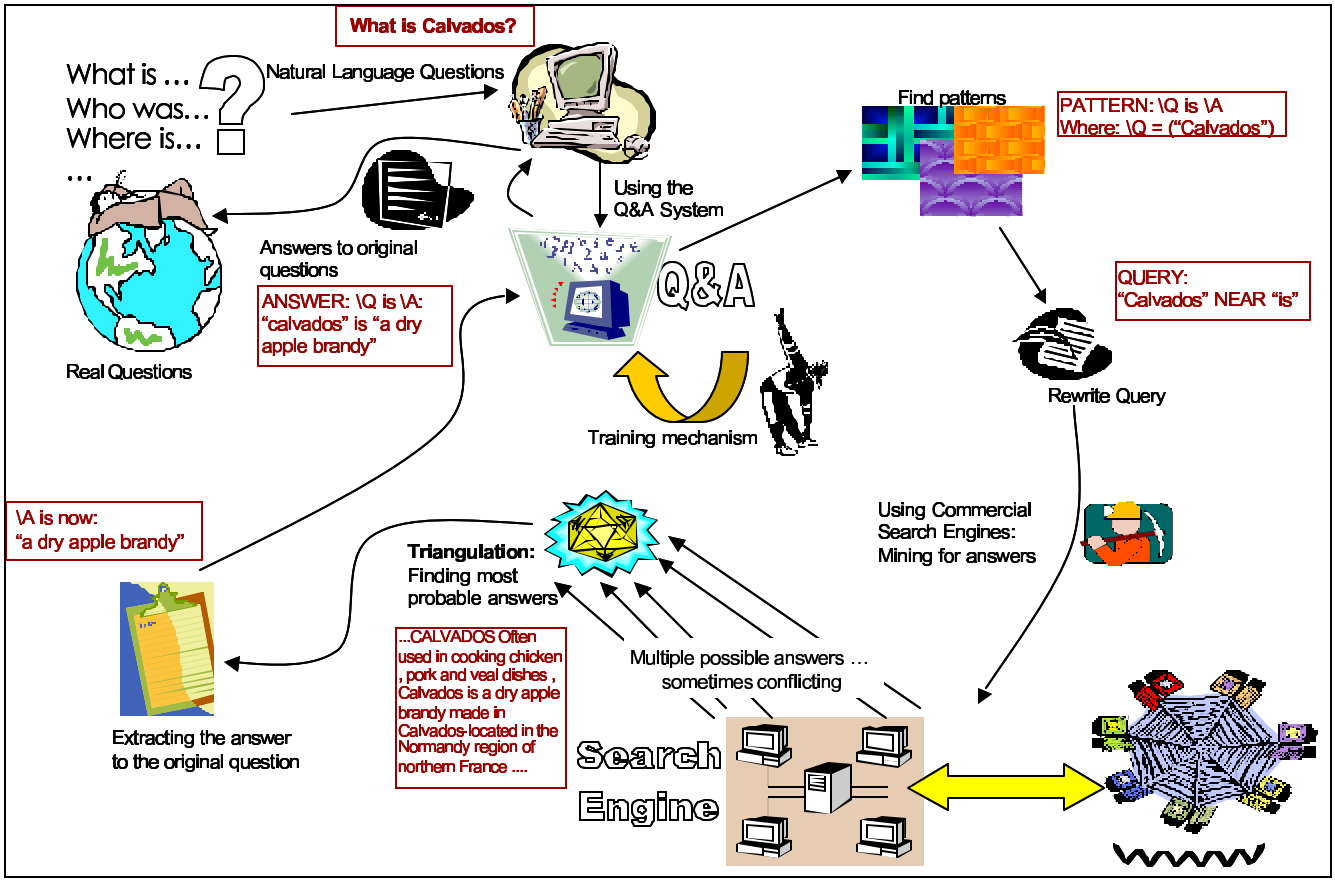


**Figure 1. The general Web QA approach**

### Processing TREC 2004 Format

TREC 2004 test questions format was drastically different from the previous competitions and resembled a sequence of questions about an explicitly stated "target", rather than independently formulated questions from earlier competitions. Since we developed and tested our system using the data and input formats from the TREC conferences prior to 2004, we run into certain challenges while interpreting TREC 2004 input and lost 14% of the score simply due to that. Our question reading algorithm (tested on the only two examples provided by NIST before the competition) simply replaced all the pronouns with the targets, if the targets or their stem were not already explicitly mentioned in the question. However, this simple approach resulted in the number of mistakes listed in Table 1. We believe all those mistakes could be avoided by simple heuristics if more examples were given in advance. We hope that the format will not be changed for the next year or number of examples comparable to the number of test questions will be provided by NIST.

| Mistake type | Examples | Number of occurrences (out of 91 questions processed) |
|---|---|---|
| Target referred via "the" determiner | What does *the name* mean or come from? Who is the sponsor of *the court*? Where is *the company* located? | 9 |
| Using both singular and plural forms for the target | Crips and Crip Gang | 1 |
| A word from another question referred via "the" determiner | Who plays *the role*? What year was *the movie* released ? | 2 |
| "Implied-only" target | How many are there now ? | 1 |

**Table 1. Mistakes in interpreting TREC 2004 format.**

### Pattern Language

Our pattern language allows any words and punctuation marks from the targeted language (English) plus the following special symbols:

*\Q* – the "question phrase" (e.g. for the question "Who is the CEO of IBM?" the \Q is "CEO of IBM"). For more complex types we use multiple question parts: *\Q1, \Q2,* etc.

***** -- a wildcard that matches any words, but not the punctuation marks;

*\p* -- any punctuation mark

\A – same matching as wildcard, but the matching text becomes a candidate answer;

\V – matches the verb from the question only (e.g. for the question "When was radio *invented*?" \V = *invented)*

Table 2 shows some examples of patterns. We believe our pattern language has more expressive power than rewrites introduced in (Dumais et al., 2002). Having wildcards also allows matching sentences that otherwise would be missed due to the presence of redundant words. E.g. "*Samuel Palmisano recently became the CEO of IBM."* matches the pattern *\A * became \Q*. This distinguishes our pattern language from those introduced earlier (Ravichandran & Hovy, 2002; Zhang & Lee, 2002).

| Question | Sentence | Pattern | Candidate Answer |
|---|---|---|---|
| What is the California's state bird? | California's state bird is the valley quail. | \s Q is \A \p | the valley quail |
| What is the capital of Taiwan? | Taipei, the capital of Taiwan, is an exciting city. | \s \A \p Q \p * \p | Taipei |
| That is the abbreviation for original equipment manufacturer? | "OEM" is the abbreviation for original equipment manufacturer. | \s \A is Q \p | "OEM" |
| What is anise? | Aniseed, also known as anise, contains several estrogenic compounds. | \s \A \p also known as Q \p * \p | Aniseed |
| What is anorexia nervosa? | Eating disorders commonly refers to anorexia nervosa, bulimia and binge-eating disorder. | \s \A refers to Q, * \p | Eating disorders commonly |

**Table 2. Examples of patterns and the matching sentences for "what is" questions.**

### Pattern Training

The purpose of training is to assign to each pattern the probability that the matching text contains a correct answer. We used the questions and correct answers from the prior TREC competitions to train our patterns. For 2004 participation, we only trained the following types: *what is/was/are, who is/was, when was, where is*, *verb-type* (e.g. Who *invented* radio? Who *manufactures* rod hockey games?), *when was born, where was born, when died,* and *when-was-verb type* (e.g. When was the radio invented?). These were the most representative questions from past TRECs. The other types would require more examples to train than is currently available.

During training, for each pair (*Question, Answer)*, the system requests the web pages from the General Purpose Search Engine (GPSE) that have both the question phrase *\Q* and the answer *\A*, preferably in proximity. We used Google for training and TREC 2004 tests. In the past, we used Alta Vista proximity operator *NEAR* to narrow down the set of pages, which was more efficient. Each sentence containing both the *\Q* and *\A* is converted into a candidate pattern by replacing the question phrase with *\Q* and the answer with *\A*. Once 200 candidate patterns are identified, each pattern is "generalized" to produce more patterns by the following :

1) replacing words (except *\A, \Q, \V*) with wildcards,
2) replacing punctuation with *\p*,
3) forming substrings containing the mentioned *\Q*, *\V* and *\A*.

Top 500 most frequent patterns after generalization are trained for the probability that the matching text includes a correct answer by modulating the pattern (as detailed below) and looking for the matches in the retrieved documents. After 40 matches from different pages have been identified, the probability is computed by the formula:

$$prob(P) = \#\,matches\ containing\ correct\ answers\ /\ \#\,total\ matches. \quad (1)$$

### Question Answering Steps

Answering the question *"Who is the CEO of IBM?"* demonstrates the steps of our algorithm.

**Type Identification.** The question itself matches the pattern *who is \Q ?*, where *\Q = "the CEO of IBM"* is the question part. Thus, the patterns corresponding to *WhatIs* type are activated for matching.

**Query modulation** converts each answer pattern (e.g. "*\A became \Q \p*") into a query for a general-purpose search engine (GPSE), e.g. *"became the CEO of IBM"*. The answer pattern *"\Q is \A"* would be converted into *"the CEO of IBM is"* etc.

**Answer Matching.** The sentence *"Samuel Palmisano recently became the CEO of IBM."* would result in a match and produce a candidate answer "*Samuel Palmisano recently*". Each candidate answer is assigned an initial score equal to the accuracy score of the matching pattern trained earlier by the formula (1).

**Answer Detailing** produces more candidate answers by forming sub-phrases from the initial candidate answers. Our sub phrases do not exceed 3 words (not counting the words from 251-word "stop word" list) and do not cross punctuation marks. In our example above, the detailed candidate answers would be *Samuel*, *Palmisano*, *recently, Samuel Palmisano, Palmisano recently*.

**The Triangulation** process establishes that *Samuel Palmisano* is the most likely answer as explained in the next section.

The algorithm stops querying GPSE when a specified number of web pages has been scanned (3000 for TREC 2004) or a specified number of candidate answers found (500 for TREC 2004). If there are fewer candidate answers found, the algorithm resorts to a "fall back" approach: it sends the question to GPSE and creates candidate answers from each sentence of the top returned 200 snippets.

### Triangulation

Triangulation, a term widely used in intelligence and journalism, stands for confirming or disconfirming facts using multiple sources (www.undp.org/eo/ADR/glossary.htm). Our triangulation algorithm can be demonstrated by the following intuitive example. Imagine that we have two candidate answers for the question *"What was the purpose of Manhattan Project?"* 1) *"To develop a nuclear bomb"* and 2) *"To create a nuclear weapon."* Those two answers reinforce (triangulate) each other since they are semantically similar. The advantage of triangulation over simple frequency counting (Dumais et al., 2002) is stronger for less "factual" questions, those that may allow variation in the correct answers. This includes such frequent types as definitions and "how to" questions.

The resulting score for each candidate answer $s^t(a)$ after triangulation is computed by the formula: $s^t(a) = \sum_{a_i \in O} s(a_i) \cdot sim(a, a_i)$, where *O* is the set of all original (before detailing) answers, *s(a)* is the original score of candidate answer *a*, and *sim(a1, a2)* is the similarity metric between answers *a1* and *a2*. Although we had tested a more fine-grained similarity metrics earlier, for the TREC competition we only used "simplified" tringulation, in which *sim(a1, a2) = 1* or *0* depending on wether stemmed string *a2* contains the stemmed string *a1* or not accordingly. This is still different from frequency count (Dumais et al., 2002) because the original probablistic score of the candidate answer is taken into consideration.

### Scalability and Responsiveness

Since TREC objective was to only test the accuracy, we were not that much concerned with the processing speed. Our Web demo finds an answer within a minute in average. The bottleneck is fetching the contents of the web pages, which can be parallelized on multiple workstations as, for example, has been successfully demonstrated in by Surdeanu et al. (2002). Another possible solution would be to have direct access to the GPSE index and cache, which may be, for example possible when the QA system is an integral part of it.

### Semantic Filtering

Contrary to many other groups involved in TREC QA, we do not use an extensive hierarchy of question types organized by the expected semantic category of an answer (e.g. person, celebrity, organization, city, state, etc.), but rather we use a small set of independently considered semantic features of candidate answers which only affect the score of a candidate answer rather than excluding it entirely. Specifically, we used the following boolean (yes/no) semantic features: *is number, is date, is year, is location, is person name* and *is proper name*. Depending on the question type we apply several heuristic rules to adjust the score of a candidate answer, for example:

```
If question starts with "who" then
        PersonNameExpected = true;
…
If PersonNameExpected then
        If not UpperCase(answer) then
                Score = Score * 0.1;
```

The features are checked by regular expressions or dictionary look-ups. We only use the lists of countries, and states to boost the matching answers for "where" questions. We had approximately a dozen of such heuristic rules. Although we set the adjustment weights manually for TREC 2004, we believe it is possible to automatically train them in future similar to training our patterns.

### Answer Projection

We did not have sufficient time to experiment with our answer projection algorithm, which we believe can be significantly improved in future. For each question and its answer, our heuristic algorithm tries to locate the most lexically similar window of text centered on the answer within the TREC collection . It works the following way. First, it retrieves the documents that both

1. Contain the answer.
2. Contain the target (or only the longest substring of the target if that substring is present in the question instead of the target, e.g. "Nirvana" instead of "group Nirvana").

If there are no documents retrieved, the algorithm moves to the next candidate answer. Once 20 answers have been unsuccessfully tried, NULL is returned as the answer. Otherwise, each retrieved document is scanned to produce the projection score by looking at 400-byte window around each occurrence of the answer. Every overlap between a word-stem from a question (excluding stopwords) and the word–stem within the window increases the projection score by the following amount:

$delta = log(N/df(t)) * (200 - distance(t)) / 200$, where

$N$ is the number of documents on the web (estimated as $3,000,000,000$ at the time of tests);

$df(t)$ is the number of documents on the web containing the word $t$ (so called *document frequency*), obtained by querying AltaVista search engine;

$distance(t)$ is the byte distance between the occurrence of the word $t$ and the occurrence of the answer.

## RESULTS AND CONCLUSIONS

Our official result on the factoid questions was 0.148. We did not attempt "list" and "other" questions. Since our projection algorithmwas not optimized and slow as a result, we unexpectedly run out of time and processed only first 92 questions. The system updated the output file in the required TREC format after each question was processed. We stopped the system shortly before the due time and submitted our single run (ASUQA). Table 3 shows the breakdown of our errors. The most significant is getting a wrong answer from the web. The second big loss is misunderstanding format of this year TREC question as discussed above. Projecting to TREC is the third significant error component.

| Error Type | #of questions | % |
|---|---|---|
| Wrong answer from Web QA | 53 | 0.58 |
| TREC format misunderstood | 13 | 0.14 |
| Unsupported (projection error) | 5 | 0.05 |
| Inexact | 2 | 0.02 |
| Correct (including NULL) | 19 | 0.21 |
| Total | 92 | 100 |

**Table 3. Breakdown of the errors.**

We estimated the average performance of all the runs submitted to TREC on the first 92 questions was only 5% better than on the rest of the question, thus the first 92 questions did not happen to be significantly more or less difficult. This allowed us to estimate our "would be" performance on all the questions by a simple linear interpolation of the number of correct answers, which we obtained to be *.21*. We find this encouraging since it would have placed us above the median.

We plan to participate in future and improve our system by possibly:

1) Making our pattern language more flexible, possibly including dependency relations produced by a parser, e.g. Minipar.
2) Find or create more training examples.
3) Convert semantic filtering heuristics into trainable rules.
4) Use combined Web and TREC collections as answer sources.
5) Add heuristics to perform anaphoric resolution within the questions.
6) Extend our triangulation mechanisms to numbers.

We always welcome suggestions from other groups and possible collabotation!

## REFERENCES


1. Agichtein, E., Lawrence, S., Gravano, L. (2001) Learning Search Engine Specific Query Transformations for Question Answering. *Proceedings of the Tenth World Wide Web Conference,* Gaithesburg, MD.
2. Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. Web Question Answering: Is More Always Better? (2002) *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland.
3. Radev, D., Fan, W., Qi, H., Wu, H., Grewal, A. (2002) Probabilistic Question Answering on the Web. *Proceedings of the 11th World Wide Web Conference*, Honolulu, HI.
4. Radev, D., Qi, H., Zheng, Z., Blair-Goldstein, S., Zhang, Z., Fan, W., and Prager, J. (2001) Mining the Web for Answers to Natural Language Questions. In the *Proceedings of the ACM CIKM 2001: Tenth International Conference on Information and Knowledge Management*, Atlanta, GA.
5. Surdeanu, M., Moldovan, D. and Harabagiu, S. (2002) Performance Analysis of a Distributed Question Answering System', *IEEE Transactions on Parallel and Distributed Systems,* 13, 6, 579-596.
6. Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for a question answering system. In Proceedings of the 40th Annual Meeting of the ACL, pages 41-47.

7. Zhang, D., and Lee, W.S. (2002). Web Based Pattern Mining and Matching Approach to Question Answering, TREC 2002.
8. Roussinov, D., and Robles, J., Learning Patterns to Answer Open Domain Questions on the Web. *In the proceedings of 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, July 25 - 29, 2004 Sheffield, UK.
9. Soubbotin, M., & Soubbotin, S. (2002). Use of patterns for detection of likely answer strings: A systematic approach. *In the Proceeding of TREC 2002*.