# Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques

**Jimmy Lin Aaron Fernandes Boris Katz Gregory Marton Stefanie Tellex**
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139
{jimmylin,adfernan,boris,gremio,stefie10}@ai.mit.edu

## Abstract

Aranea is a question answering system that extracts answers from the World Wide Web using knowledge annotation and knowledge mining techniques. Knowledge annotation, which utilizes semistructured database techniques, is effective for answering large classes of commonly occurring questions. Knowledge mining, which utilizes statistical techniques, can leverage the massive amounts of data available on the Web to overcome many natural language processing challenges. Aranea integrates these two different paradigms of question answering into a single framework. For the TREC evaluation, we also explored the problem of answer projection, or finding supporting documents for our Web-derived answers from the AQUAINT corpus.

## 1 Introduction

Aranea, MIT's entry to the TREC Question Answering track, focused on extracting answers from the World Wide Web. Our system was organized around a modular framework that integrates two different paradigms of question answering: knowledge annotation using annotated structured and semistructured resources and knowledge mining using statistical techniques that leverage data redundancy (cf. (Lin and Katz, 2003)).

Aranea's approach to question answering is motivated by an observation about the empirical distribution of user queries, which quantitatively obey Zipf's Law—a small fraction of question types account for a significant portion of all question instances. Certain natural language questions tend to occur much more frequently than others, an observation that is confirmed by many different sources: TREC queries (Lin, 2002), logs from the START question answering system (Katz, 1997), and logs of a commercial search engine (Lowe, 2000). Furthermore, many questions ask for the same type of information and differ only in the specific object questioned, e.g., "What is the population of (the United States, Mexico, Canada,...)?" We can group these questions into a single class (or type), i.e., "What is the population of $x$?" where $x$ can be any country, and find the answer in a database. *Knowledge annotation* is a question answering strategy that allows heterogeneous sources on the Web to be accessed as if it were a uniform database. By connecting natural language queries to this "virtual" database, Aranea can answer large classes of commonly-occurring questions.

Typically, Zipf curves have broad tails where individual instances are either unique or account for an insignificant fraction of total instances. This observation also holds true for the distribution of user questions: in addition to asking large classes of commonly-occurring questions, users also pose a significant number of unique questions that cannot be easily classified into common categories or grouped by simple patterns, e.g., "What format was VHS's main competition?" To answer these questions, Aranea employs what we call redundancy-based *knowledge mining* techniques.

For the TREC evaluation, the extraction of answers from the Web necessitated an extra step in the question answering process, usually known as *answer projection*. For every Web-derived answer, our system had to find a supporting document from the AQUAINT corpus, even though the corpus itself was never used in the question answering process. This additional component had a significant impact on the overall performance of our system.

## 2 Overall Framework

The general architecture of the Aranea system is shown in Figure 1. User questions are routed to two separate components, one that employs the knowledge annotation strategy (Section 3) and one that utilizes the knowledge mining strategy (Section 4). Both components consult the World Wide Web to generate candidate answers, and the results of both components are piped through a knowledge boosting and cleanup module (Section 5), which check the answer candidates against a number of heuristics to ensure their validity. Finally, the answer projection module (Section 6) finds an article from the AQUAINT corpus that adequately supports the answer derived from the Web.

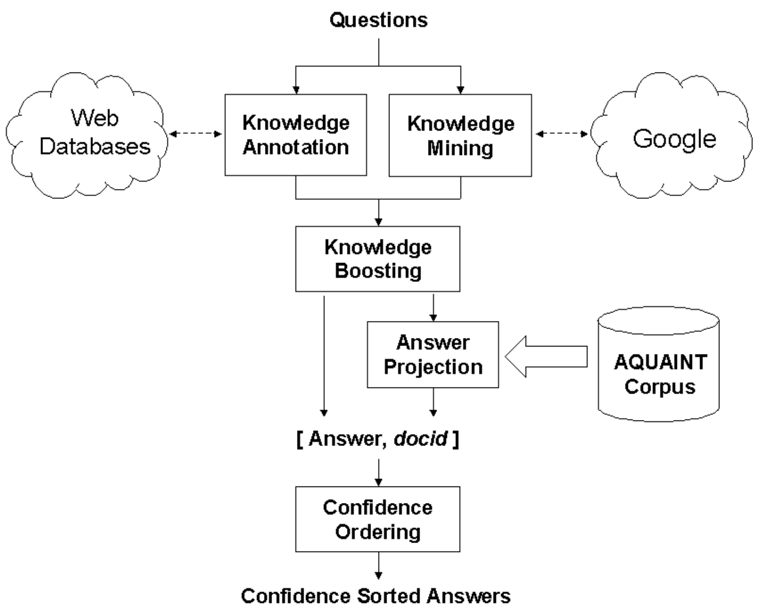

Figure 1: Overall Architecture of the Aranea question answering system.

Aranea supports a modular pipeline architecture by enforcing input and output constraints at the module interfaces. The input and output of each module is an XML-encoded data structure that keeps track of the current computational state. Aranea modules are conceptualized as transformations over this XML data structure.

## 3 Knowledge Annotation

Although the Web consists largely of unorganized pages, pockets of structured and semistructured knowledge exist as valuable resources for question answering. For example, the CIA World Factbook provides political, geographic, and economic information about every country in the world; 50states.com contains numerous properties related to US states from state bird to land area; Biography.com has collected profiles of over twenty-five thousand famous (and not-so-famous) people; the Internet Movie Database stores entries for hundreds of thousands of movies, including information about their cast, production staff, and dozens of other properties.

To effectively use these existing resources for question answering, the plethora of knowledge sources must be integrated, or federated, under a common interface or query language. Database concepts and techniques provide the tools to accomplish just that. In fact, since many of these sources are part of the "deep" or "invisible" Web, they are inaccessible to search engines and can only be modeled as "virtual" databases. We have developed a schema-based technique called knowledge annotation by which natural language queries can be connected to semistructured knowledge sources.

Our knowledge annotation strategy provides an effective mechanism for answering natural language questions. Because users frequently ask the same types of questions, a few well-chosen knowledge sources are sufficient to provide good knowledge coverage. For example, we have verified that ten Web sources can provide answers to 27% of TREC-9 and 47% of TREC-2001 questions from the QA track (Lin, 2002). In addition, other researchers (Hovy et al., 2002) have noticed the importance of external knowledge sources for question answering.

The knowledge annotation component of Aranea is a simplified implementation of the system used by the START (Katz, 1988; Katz, 1997) and Omnibase (Katz et al., 2002a; Katz et al., 2002b) systems. START is a natural language understanding system, and Omnibase is a virtual database that provides uniform access to heterogenous and distributed Web sources via a wrapper-based framework. A simplified version of natural language annotation technology (Katz, 1997) is employed in database access schemata to mediate between natural language and database queries.

Since it came on-line in December 1993, START has engaged in exchanges with hundreds of thousands of users all over the world, supplying them with useful knowledge. However, because the system provides users with paragraph-sized answers that often contain multimedia fragments such as pictures and audio clips, they are not suitable for a TREC-style evaluation. There is evidence to support that

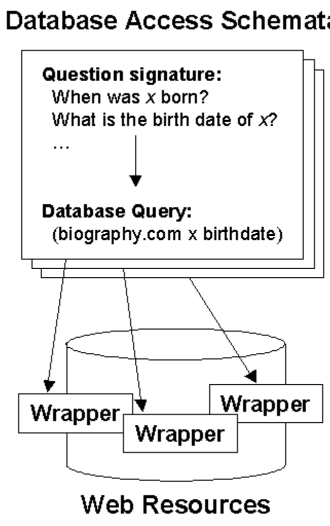

Figure 2: The knowledge annotation component of Aranea

paragraph-sized chunks form the most suitable unit of response to a user question, because it provides not only the exact answer, but additional contextual information that may help with interpretation and analysis (Lin et al., 2003). Because this year's TREC QA track accepted only exact answers, we found it inappropriate to directly evaluate START and Omnibase.

### 3.1 Database Access Schemata

A collection of database access schemata and wrappers comprise the knowledge annotation component of Aranea (Figure 2). Each schema is composed of two connected parts: the question signature and the database query. A question signature is a collection of regular expressions that match a specific class of user questions, e.g., requests for birth dates of people.[1] These patterns are paired with unfilled database queries that are dynamically instantiated with bindings extracted from the question signature. Consider a typical database access schema:

When was $x$ born?
What is the birthdate of $x$?
...
$\rightarrow$ `(biography.com` $x$ `birthdate)`

In this example, questions that ask for the birth dates of various people are translated into an *object–property–value* database query (Katz et al., 2002a). These queries specify the data source where the answer could be found (`biography.com`), the *object* in question ($x$), and the *property* sought after (`birthdate`). The *value* of the object's property typically answers the user's question.

The knowledge annotation component of Aranea operates by matching the user question against schemata stored in the knowledge base and executing database queries generated by the matched schemata.

The Aranea database engine is responsible for retrieving the actual answers by executing the database queries. The retrieval of such information depends on the type of the data source: Some sources are stored locally, and may translate into a file lookup. Other sources are stored on remote Web sites behind a CGI interface; executing database queries on these sources requires dynamically reconstructing an HTTP request and properly parsing the resulting HTML document. More details on the process of structuring a knowledge source for database access is discussed in the next section.

### 3.2 Knowledge Engineering

Teaching Aranea's knowledge annotation component to answer different classes of natural language questions involves three separate steps borrowed from START and Omnibase: identifying question classes and knowledge sources, writing the database access schemata, and writing wrappers for the data sources.

The first step in the knowledge engineering process is to identify the class of questions to be answered and an adequate knowledge source that provides the answer. Empirical analysis of the question distribution provides hints on the effectiveness of any effort. We have noticed, for example, that users frequently asked about the demographics and economics of countries. These questions can be answered by writing a schema that uses information found within the CIA World Factbook.

Once a question class and a knowledge source have been determined, regular expression patterns that capture the general form of the question must be written. Usually, such patterns take into account various alternative formulations of the same query. These patterns must clearly indicate the noun phrase that can be parameterized. For example, in the question class "What is the GDP of $x$?", $x$ stands as a generic placeholder for country names. The mapping between the natural language patterns and the database query must also be specified, e.g., the $x$ extracted from the previous question pattern fills the $x$ slot in the database query `(cia-factbook` $x$ `gdp)`

After a database access schema has been crafted, a wrapper must be written for the knowledge resource. These wrappers supply the actual procedures used to execute queries of a specific form. Although

[1]These question signatures are a simplified version of natural language annotations used by START, which are parsed into and stored as ternary expressions. Because matching occurs at the level of the parsed structures, powerful linguistic machinery can be employed to handle different linguistic phenomena, e.g., synonymy, hyper/hyponymy, syntactic alternations, etc.

Aranea provides a generic framework for organizing the queries and convenient libraries of often-used functionality, specific implementations for accessing various data sources must be provided separately. Typically, executing a query involves either looking up the information in a locally stored database (ranging in complexity from tab-delimited flat files to full SQL databases), or executing a CGI request to retrieve a dynamically generated page from a remote Website and performing additional postprocessing to extract only the relevant fragments.

The Aranea system deployed for the TREC competition included twenty-eight schemata that access seven different data sources. Here are two examples:

- `Biography.com` This source provides information about the lifespan, birth dates, and death dates of various well-known people. Answering questions about such properties involves dynamically retrieving pages from `biography.com` (via CGI) and performing simple pattern matching on the HTML document to extract exact dates.
- `CIA World Factbook` This resource provides various useful facts about countries, e.g., population, area, capital, etc. This information was download and structured in a locally-stored tab-delimited file. Questions about various properties of world countries are translated into simple file lookups.

## 4 Knowledge Mining

The knowledge mining approach to question answering is based on the observation that as the size of a text collection increases, occurrences of the correct answer tend to also increases. Specifically, Breck *et al.* (Breck et al., 2001) noticed a correlation between the number of times an answer appeared in the TREC corpus and the average performance of TREC systems on that particular question. This result verifies intuition: the more times an answer appears, the easier it is to find it. The knowledge mining component of Aranea extends this insight to the World Wide Web, and leverages the Web's massive size for question answering.

As a text collection, the Web is larger in size than any research corpus by several orders of magnitude. An important implication of this size is the amount of data redundancy inherent in the text collection; potentially, each item of information has been stated in a variety of ways, in different documents. However, data redundancy is counterbalanced by the poor quality of individual documents.

A question answering system can utilize massive amounts of Web data in two ways: as a surrogate for sophisticated natural language techniques and as a method for overcoming poor document quality. Consider the question "When did Wilt Chamberlain score 100 points?" Here are two possible answers:

> (1) Wilt Chamberlain scored 100 points on March 2, 1962 against the New Yorks Knicks.
>
> (2) On December 8, 1961, Wilt Chamberlain scored 78 points in a triple overtime game. It was a new NBA record, but Warriors coach Frank McGuire didn't expect it to last long, saying, "He'll get 100 points someday." McGuire's prediction came true just a few months later in a game against the New York Knicks on March 2.

The answer could be more easily extracted from sentence (1) than from passage (2). In general, the task of answering a question is not very difficult if the document collection contains the answer stated as a simple reformulation of the question. In these cases, simple techniques, e.g., keywords or regular expressions suffice to achieve state-of-the-art performance. As the size of the document collection grows, the more likely it is that question answering systems can find statements that answer the question by matching a simple reformulation.

Without the luxury of massive amounts of data, a question answering system may be forced to extract answers from passages in which they are not obviously stated, e.g., passage (2). In these cases, sophisticated natural language processing may be required to relate the answer to the question, e.g., recognizing syntactic alternations, resolving anaphora, making commonsense inferences, performing relative date calculations, etc.

The World Wide Web is so big that simple pattern matching techniques can often replace the need to understand both the structure and meaning of language. The answer to a question could be extracted by searching directly for an anticipated answer form, e.g., in the above example, by searching for the string "Wilt Chamberlain scored 100 points on" and extracting words occurring to the right. Naturally, this simple technique depends crucially on the corpus having an answer formulated in a specific way. Thus, the larger the text collection is, the greater the probability that simple pattern matching techniques will yield the correct answer. Data redundancy enables a simple trick to overcome many troublesome issues in natural language processing, e.g., alternations, anaphora, etc.

Despite the apparent advantages of massive amounts of data, the process of answering questions using the Web is complicated by the low average quality of individual documents. Due to the low barrier of entry in Web publishing, many documents are poorly written, barely edited, or simply contain incorrect information. As a result, text extracted from a single document cannot be trusted as the correct answer. This problem can also be alleviated through data redundancy. A single instance of a candidate answer may not provide sufficient justification, but multiple occurrences of the same answer in different documents lends credibility to the proposed answer.[2]

The tremendous amounts of information on the World Wide Web would be useless without an effective method of data access. Providing the basic infrastructure for indexing and retrieving text at such scales is a tremendous engineering task. Fortunately, such services already exist, in the form of search engines. For example, Google, the largest of the Web search engines, boasts over 3 billion documents in its index.[3] Using existing search engines as information retrieval backends, we can focus our efforts on answer extraction.

Many of the knowledge mining techniques described above have been implemented in previous systems (Brill et al., 2001; Buchholz, 2001; Clarke et al., 2001; Kwok et al., 2001; Soubbotin and Soubbotin, 2001; Brill et al., 2002). The introduction of redundancy-based question answering using the Web (Brill et al., 2001) at last year's TREC conference has generated a new set of techniques for attacking the question answering problem. We have taken advantage of previous experiences to refine many techniques within a better engineered framework. In particular, our infrastructure supports a modular architecture that allows specific functionality to be encoded into manageable components. This not only allows for faster development cycles, but facilitates glass-box testing to properly determine the effectiveness of various techniques.

The data flow in the knowledge mining component of Aranea is shown in Figure 3. In the following sections, we describe each module in detail. Each module accepts an Aranea XML data structure as input and returns a structure of the same type as output; the functionality of each module is implemented as internal transformations on the XML data structure.

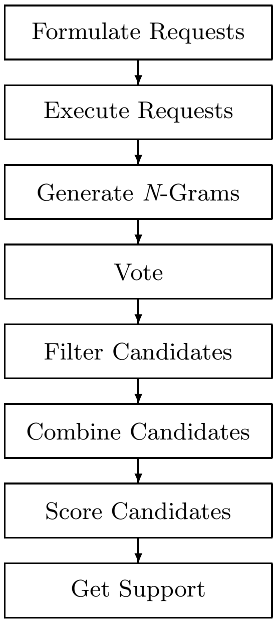

Figure 3: Data flow in the knowledge mining component of Aranea

### 4.1 Formulate Requests

The first step in answering natural language questions in the knowledge mining component is to translate them into Aranea queries, or requests. These requests specify the textual context in which answers are likely to be found, and are analogous to queries posed to information retrieval systems. However, because Aranea relies on Web search engines to fulfill these requests, fine-grained control over the result set is impossible. Aranea instead relies on *quantity* to make up for lack of *quality*.

Two types of queries are generated by this module: *exact* (or *reformulation*) queries and *inexact* (or *backoff*) queries. Queries of both types are concurrently generated, but usually given different scores.

An inexact query indicates that an answer is likely to be found within the vicinity of a set of keywords. They are composed by treating the natural language question as a bag of words.

An exact query specifies the location of a potential answer in more detail, e.g., the answer to "When did the Mesozoic period end?" is likely to appear within ten words and fifty bytes to the right of the exact phrase "the Mesozoic period ended". Exact queries in Aranea are generated by approximately a dozen pattern matching rules based query terms and their part-of-speech tags; morpho-lexical pattern matches trigger the creation of reformulated exact queries. As an example, the previous query was generated by the rule "*wh-word* did ... *verb* $\rightarrow$ ... *verb*+ed". An internal lexicon ensures that the generated verb remains properly inflected.

As a complete example, the requests generated in response to the question "When did the Mesozoic period end?" are shown in Figure 4. Aranea generates two inexact and one exact requests; each request is assigned a basic score, which establishes the relative importance of the queries.

### 4.2 Execute Requests

The request execution module is responsible for retrieving textual "snippets" that honor the constraints set forth in each request. Currently, the Google search engine is used to mine text from the

[2]Unfortunately, this technique equates the *most popular* answer with the *correct* answer, which occasionally results in very comical responses.

[3]as of early 2003

**Query:** When did the Mesozoic period end
  Type: inexact
  Score: 1
  Number of snippets to mine: 100

**Query:** the Mesozoic period ended
  Type: inexact
  Score: 1
  Number of snippets to mine: 100

**Query:** the Mesozoic period ended ?x
  Type: exact
  Score: 2
  Number of snippets to mine: 100
  Maximum length for `?x`: 50
  Maximum word count for `?x`: 5

Figure 4: Typical requests generated by Aranea.

Web. In the case of inexact requests, the entire summary provided by Google is extracted for further processing. For exact queries, the request execution module performs additional pattern matching to ensure that the correct positional constraints are satisfied.

### 4.3 Generate *N*-Grams

This module exhaustively generates all possible unigrams, bigrams, trigrams, and tetragrams from the text fragments generated by the request execution module. These *n*-grams, which are given initial scores equal to the weight of the request from which they derive, serve as the raw candidate answers.

### 4.4 Vote

The voting module collates the *n*-grams generated by the previous module. The new score of each answer candidate is equal to the sum of the scores of all occurrences of that particular *n*-gram. This module has the effect of promoting text fragments that occur frequently (in the context of query terms), and are hence more likely to answer the user question.

### 4.5 Filter Candidates

In this stage of the processing, a coarse-grained filter in applied to answer candidates:

- Candidates that begin or end with stopwords are discarded.
- Candidates that contain words found in the user question are discarded. The only exception to this rule is question focus words, e.g., a question beginning with "how many meters..." can be answered by an expression containing the word *meters*.

In addition, this stage encodes a few heuristics that can potentially decrease the number of answer candidates. For example, the answer to "how far", "how fast", "how tall", etc., questions must contain a numeric component (either numeric digits or numerals); thus, we can safely discard all answer candidates that do not fit these criteria. We have also noticed that "who" and "where" questions usually cannot be answered with expressions that contain tokens consisting of numeric digits; Aranea can similarly reduce the number of answer candidates based on this criterion. The general principle embodied in this module is to filter with high confidence, erring on the side of being too lenient. False positives can always be sorted out by later modules, but the system will not be able to recover from false negatives.

### 4.6 Combine Candidates

In this module, shorter answers are used as evidence to boost the score of longer answers. If a portion of a candidate answer appears itself as a candidate answer, then the score of the shorter answer is added to the score of the longer answer. For example, if "de Soto" appears on the list of candidate answers along with "Hernando de Soto", the score of the shorter candidate would be added to the score of the longer one. This module counteracts the tendency of the *n*-gram generation and voting modules to favor shorter answers.

### 4.7 Score Candidates

The score of each answer candidate is multiplied by the following factor:

$$\frac{1}{|A|}\sum_{w\in A}\log(\frac{N}{w_c})$$

$A$ is a set of keywords in the candidate answer; $N$ is the total number of words in the AQUAINT corpus; $w_c$ is the number of occurrences of word $w$ in the AQUAINT corpus. Each answer candidate is scaled by the average of an *idf*-like value of its component keywords. This scoring balances the effect of individual keywords having different (unconditioned) priors. Since the exact distribution of unigrams on the Web can not be easily obtained in a reliable manner, Aranea uses statistics from the AQUAINT corpus as a surrogate.

### 4.8 Get Support

This module performs a final sanity check on the candidate answers. It verifies that final candidate answers actually appear in the original text snippets mined from the Web. Occasionally, the various modules within the knowledge mining component of the system will assemble a nonsensical answer; this module ensures that such answers are discarded.

## 5 Answer Boosting and Cleanup

Results from both the knowledge annotation and knowledge mining components of Aranea are subjected to a series of heuristic checks. These heuristics may employ external knowledge resources to verify the candidate answers.

The answer boosting module of Aranea contains heuristics specifically dedicated to verifying geographic locations. We have gathered large lists of known geographic entities, e.g., world cities, US cities, etc.; these lists allow us to "boost" the score of certain answer candidates in response to "what city", "what state", "what country", "what province", etc. questions.

Questions requiring dates as answers similarly receive special treatment. Named entity detectors allow us to promote dates over other noun phrases. Knowledge of dates also helps Aranea extract the exact answers. For example, a candidate answer to a "what year" question often contains extra information such as the month and day; Aranea removes such extraneous information.

Beyond a few simple heuristics, Aranea also performs part-of-speech tagging on the answer candidates to ensure that they are full constituents (NP or VP). Extra leading or trailing words are trimmed.

## 6 Answer Projection

The final step in the preparation of an answer derived either from knowledge annotation or knowledge mining is answer projection, during which each Web-extracted answer is paired with a document from the AQUAINT corpus to form the basic [*answer*, *docid* ] response unit. Answer projection was accomplished in a two step process: first, a set candidate documents was gathered; then, a modified passage retrieval algorithm scanned the documents to pick the best document.

We experimented with three different methods of retrieving a candidate set of documents on which to project our Web-derived answers:

- **NIST documents.** The top fifty documents supplied by NIST served as the baseline set of candidate documents for answer extraction.
- **MultiText passages.** We have implemented the passage retrieval algorithm described by Clarke *et al.* (2000). A set of passages generated by this algorithm serves as the candidate documents for answer projection.
- *PC3* **MultiText passages.** We have augmented the MultiText passage retrieval algorithm by a backoff procedure we call *pc3*. Our algorithm applies a series of controlled query expansion loops, which successively broadens the query terms (e.g., by including different inflections and synonyms of the keywords) until an adequate set of candidate passages have been found.

After a set of candidate documents has been gathered, the answer projection module applies a modified window-based passage retrieval algorithm to score the documents. Each 140-byte window is given a score equal to the number of times keywords from both the question and candidate answer appears, with the restriction that at least one keyword from the question must appear in the particular passage. The score of a document is simply the score of the highest scoring passage. The highest scoring document is paired with the Web-derived candidate answer as the final response unit.

## 7 Confidence Ordering

This year's TREC evaluation required participants to sort answers according to confidence, motivated by the importance of a system knowing when it is likely to be right or wrong. Although this was certainly an interesting aspect of the question answering task, due to time constraints, we were unfortunately not able to devote much attention to it.

For the deployed TREC system, we employed a crude algorithm:

- All *when* questions were placed before all *who* and *where* questions, which were ordered before all *what* questions. All other questions were placed after. We discovered through ad-hoc experimentation that Aranea generally performed better on certain types of questions; the confidence ordering reflected our experiences.
- Within each type of question, answers derived from knowledge annotation were always placed before answers derived from knowledge mining.
- Answers were sorted by the document score produced by the answer projection algorithm
- Any further ties were broken by scores generated by the knowledge mining component.

## 8 Results

The official TREC results are shown in Table 1. The only difference between our three runs was the method used to generate the initial set of candidate documents for answer projection:

- `aranea02a` used only the top fifty NIST-supplied documents.
- `aranea02pbq` used the top fifty NIST-supplied documents and passages derived from the MultiText algorithm.
- `aranea02pc3` used the top fifty NIST-supplied documents and passages derived from the *pc3* variant of the MultiText algorithm.

In addition, we analyzed Aranea's performance without taking into account answer projection. We felt that this particular instance of answer projection is an artifact of the TREC evaluation, and not

| | | aranea2002a | | aranea2002pbq | | aranea2002pc3 | | aranea2002 | |
|---|---|---|---|---|---|---|---|---|---|
| | | NIST Docs | | MultiText | | *pc3* MultiText | | no projection | |
| **Knowledge Annotation** | correct | 22 | 4.4% | 23 | 4.6% | 22 | 4.4% | 30 | 6.0% |
| | inexact | 2 | 0.4% | 3 | 0.6% | 2 | 0.4% | 2 | 0.4% |
| | unsupported | 8 | 1.6% | 6 | 1.2% | 8 | 1.6% | - | - |
| | wrong | 10 | 2.0% | 10 | 2.0% | 10 | 2.0% | 10 | 2.0% |
| | **total** | 42 | 8.4% | 42 | 8.4% | 42 | 8.4% | 42 | 8.4% |
| **Knowledge Mining** | correct | 130 | 26.0% | 131 | 26.2% | 129 | 25.8% | 153 | 30.6% |
| | inexact | 34 | 6.8% | 33 | 6.6% | 34 | 6.8% | 43 | 8.6% |
| | unsupported | 32 | 6.4% | 32 | 6.4% | 32 | 6.4% | - | - |
| | wrong | 262 | 52.4% | 262 | 52.4% | 262 | 52.4% | 262 | 52.4% |
| | **total** | 458 | 91.6% | 458 | 91.6% | 458 | 91.6% | 458 | 91.6% |
| **Total** | correct | 152 | 30.4% | 154 | 30.8% | 151 | 30.2% | 183 | 36.6% |
| | inexact | 36 | 7.2% | 36 | 7.2% | 36 | 7.2% | 45 | 9.0% |
| | unsupported | 40 | 8.0% | 38 | 7.6% | 40 | 8.0% | - | - |
| | wrong | 272 | 54.4% | 272 | 54.4% | 273 | 54.6% | 272 | 54.4% |
| | **total** | 500 | 100% | 500 | 100% | 500 | 100% | 500 | 100% |
| | CWS score | **0.433** | | **0.427** | | **0.421** | | **0.529** | |

Table 1: TREC Results

| | | aranea2002a | aranea2002pbq | aranea2002pc3 | aranea2002 |
|---|---|---|---|---|---|
| | | NIST Docs | MultiText | *pc3* MultiText | no projection |
| **Knowledge Annotation** | correct | 52.4% | 54.8% | 52.4% | 71.4% |
| | inexact | 4.8% | 7.1% | 4.8% | 4.8% |
| | unsupported | 19.0% | 14.3% | 19.0% | - |
| | wrong | 23.8% | 23.8% | 23.8% | 23.8% |
| **Knowledge Mining** | correct | 28.4% | 28.6% | 28.2% | 33.4% |
| | inexact | 7.4% | 7.2% | 7.4% | 9.4% |
| | unsupported | 7.0% | 7.0% | 7.0% | - |
| | wrong | 57.2% | 57.2% | 57.4% | 57.2% |

Table 2: Performance of individual components.

inherent in the question answering task itself. We rescored the unsupported judgments of `aranea02a` either as inexact or correct, careful to adhere to the same standards of judgment as the other runs. This result is shown in the last column of Table 1.

In the formal TREC runs, our system answered approximately thirty percent of the questions correctly. Disregarding answer projection, Aranea provided exact, correct answers for nearly thirty-seven percent of the questions. Out of five hundred questions, 42 (8.4%) answers were contributed by Aranea's knowledge annotation component; the knowledge mining component accounted for the rest, or 458 (91.6%) questions.

Approximately 15% of answers judged as correct were derived from knowledge annotation techniques. We believe that this performance is remarkable, considering that our system contained only twenty-eight data access schemata over seven sources, representing no more than a few person-days worth of knowledge engineering effort. Our experiences with START and Omnibase have helped us streamline the knowledge engineering process, allowing us to rapidly structure knowledge sources to answer English questions. These results also verify that analysis of the typical distribution of user questions can help guide the knowledge engineering effort. Our database access schemata were geared towards answering the most frequently occurring questions from the previous TREC evaluations; many of the same question types also appeared in this year's evaluation.

Overall, we noticed that answer projection was the obvious weak link in Aranea. For approximately twenty percent of our Web-derived answer, our system was unable to find an adequate supporting document, which resulted in a drastic reduction of our overall TREC score. Our passage retrieval algorithm was not sophisticated enough to ignore documents that contained keywords from the question and answer, but in fact did not answer the question. More future research is required to obtain better answer projection performance.

Individual analysis of each Aranea component is shown in Table 2. In general, the database component achieves much higher accuracy than the knowledge mining component, due to the knowledge engineering effort involved in creating database access schemata. However, projecting answers derived from database access appears more difficult than answers derived from knowledge mining. Once again, we believe that Aranea demonstrates the validity and effectiveness of knowledge engineering in the question answering process. Knowing when to apply manual effort and selectively using human labor can translate into a big payoff in terms of performance enhancement.

## 9 Contributions

The Aranea system presents two different paradigms for approaching the question answering problem. In the knowledge annotation approach, natural language questions can be translated into database queries, which then extract answers from the Web. In the knowledge mining approach, data redundancy on the Web can be leveraged to overcome many difficult problems in natural language processing.

Aranea smoothly integrates both the knowledge annotation and knowledge mining approach into a uniform framework. With knowledge about the types of questions that users ask, we were able to utilize each paradigm effectively.

Another insight we gained in developing Aranea is to let the analysis of user questions guide our knowledge engineering effort. By correctly anticipating the types of questions users typically ask, we were able to construct effective database access schemata with reasonable amounts of manual labor.

We believe that Aranea provides a well-engineered platform for experimenting with various Web-based question answering techniques. In the future, we will continue to refine existing technology and develop new methods for answering natural language questions.

## 10 Acknowledgements

This research is funded by DARPA under contract number F30602-00-1-0545 and administered by the Air Force Research Laboratory. Additional funding is provided by the Oxygen Project.

## References

Eric Breck, Marc Light, Gideon S. Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01) Workshop on Open-Domain Question Answering.*

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).*

Sabine Buchholz. 2001. Using grammatical relations, answer frequencies and the World Wide Web for question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

Charles Clarke, Gordon Cormack, Derek Kisman, and Thomas Lynam. 2000. Question answering by passage selection (multitext experiments for TREC-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9).*

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2001).*

Eduard Hovy, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2002. Using knowledge to facilitate factoid answer pinpointing. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002).*

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. 2002a. Omnibase: Uniform access to heterogeneous data for question answering. In *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002).*

Boris Katz, Jimmy Lin, and Sue Felshin. 2002b. The START multimedia information system: Current technology and future directions. In *Proceedings of the International Workshop on Multimedia Information Systems (MIS 2002).*

Boris Katz. 1988. Using English for indexing and retrieving. In *Proceedings of the 1st RIAO Conference on User-Oriented Content-Based Text and Image Handling (RIAO '88).*

Boris Katz. 1997. Annotating the World Wide Web using natural language. In *Proceedings of the 5th*

*RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97)*.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the Web. In *Proceedings of the Tenth International World Wide Web Conference (WWW2001)*.

Jimmy Lin and Boris Katz. 2003. Question answering techniques for the World Wide Web. In *EACL-2003 Tutorial*.

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. The role of context in question answering systems. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems (CHI 2003)*.

Jimmy J. Lin. 2002. The Web as a resource for question answering: Perspectives and challenges. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002)*.

John B. Lowe. 2000. What's in store for question answering? (invited talk). In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*.

Martin M. Soubbotin and Sergei M. Soubbotin. 2001. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.