Integrating Web-based and Corpus-based Techniques for Question Answering

Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt,
Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, Federico Mora
MIT Computer Science and Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139

1 Introduction

MIT CSAIL's entry to this year's TREC Question Answering track focused on merging answers mined from the World Wide Web with answers extracted from the AQUAINT corpus of news articles. Our multi-source approach to question answering necessitates tight integration of different techniques that capitalize on different characteristics of the Web and closed corpora.

The advantages that the Web provides for question answering are well known and have been exploited in previous systems (Brill et al., 2001; Clarke et al., 2001; Dumais et al., 2002; Lin et al., 2002). The immense amount of freely available unstructured text provides data redundancy, which can be leveraged with data-driven techniques. In many ways, we can utilize huge quantities of data to overcome many thorny problems in natural language processing such as lexical ambiguity and paraphrases. Furthermore, Web search engines such as Google provide a convenient front end for accessing and filtering enormous amounts of Web data. We have identified this class of techniques as the *knowledge mining* approach to question answering (Lin and Katz, 2003).

In addition to viewing the Web as a repository of unstructured documents, we can also leverage pockets of structured and semistructured sources available on the Web using *knowledge annotation* techniques (Katz et al., 2002; Lin and Katz, 2003). These sources can be employed to directly answer factoid questions or "distilled" into knowledge bases to assist in answering list and definition questions.

While the Web is undeniably a useful resource for question answering, it is not without drawbacks. Useful knowledge on the Web is often drowned out by the sheer amount of irrelevant material, and statistical techniques are often insufficient to separate right answers from wrong ones. Outstanding issues in question answering include anaphora, paraphrases, temporal expressions, lexical ambiguity, and reasoning. Solving these problems will require major advances in language processing capabilities. Furthermore, the setup of the TREC evaluations necessitates an extra step in the question answering process for systems that extract answers from the Web, typically known as *answer projection*. For every Web-derived answer, a system has to find a supporting document from the AQUAINT corpus, even though the corpus itself may not have ever been used in the question answering process.

We attempt to address some of the above issues by integrating Web-based question answering techniques with more traditional corpus-based techniques driven by information retrieval and information extraction technology. The dominant paradigm in question answering over the last few years has been to employ document retrieval to first narrow down the corpus to a candidate set of documents, and then apply named-entity extraction technology to identify phrases that match the expected answer type derived from the original question. Previously, our entries to the TREC evaluation exclusively relied on Web-based techniques (Lin et al., 2002); however, corpus- and Web-based strategies should play complementary roles in an overall question answering framework.

2 List Questions

For answering list questions, our system employs a traditional pipeline architecture with distinct states for document retrieval, passage retrieval, answer extraction, and duplicate removal (see Figure 1). The general idea is to successively narrow down the AQUAINT corpus to manageable-sized passages, and then employ knowledge of fixed lists to extract relevant answers. The following subsections describe this process in greater detail.

2.1 Document Retrieval

In response to a natural language question, our document retriever provides a set of candidate documents that are likely to contain the answer; these documents serve as the input to additional processing modules. As such, the importance of document retrieval cannot be underestimated: if no relevant documents are retrieved, any amount of additional processing would be useless.

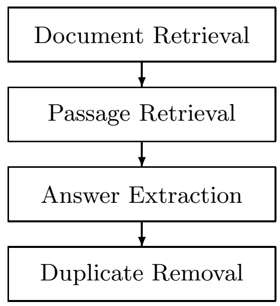

Figure 1: Architecture for answering list questions.

For our document retriever, we relied on Lucene, a freely available open-source IR engine.[1] Lucene supports a weighted boolean query language, although it performs ranked retrieval using a standard *tf.idf* model. We have previously discovered that for the purpose of passage retrieval, Lucene performs on par with state-of-the-art probabilistic systems based on the Okapi weighting (Tellex et al., 2003).

One way to boost document retrieval performance is to employ query expansion techniques. In our TREC entry this year, we implemented two separate query generators that take advantage of linguistic resources to expand query terms. Lucene provides a structured query interface that gives us the ability to fine tune our query expansion algorithms.

2.1.1 Method 1

Our first query generator improves on a simple bag-of-words query by taking inflectional and derivational morphology into account: queries are a conjunction of disjuncts, where each disjunct contains morphological variants of the keywords. Base query terms are extracted from a natural language question after dropping all stopwords. Assuming we have three query terms, A, B, and C, arranged in increasing *idf*, our first query method would generate the following queries:

$$\begin{array}{l} A \wedge B \wedge C \\ e(A) \wedge e(B) \wedge e(C) \\ e(B) \wedge e(C) \\ e(C) \\ e(A) \wedge e(B) \\ e(B) \\ e(A) \end{array}$$

where

$$e(x) = x \vee \text{inflect}(x)^{0.75} \vee \text{derive}(x)^{0.50}$$

where $\text{inflect}(x)$ and $\text{derive}(x)$ indicate inflectional and derivational morphological forms of x, respectively. The first query is simply a conjunction of all non-stopwords from the question. The second query is a conjunction where each of the conjuncts is a disjunct of morphologically expanded query terms. Inflection variants are generated with the assistance of WordNet to handle irregular forms. Derivational variants are generated by a version of CELEX that we manually annotated. Using Lucene's query weighting mechanism, inflected forms are given a weight of 0.75, and derivational forms a weight of 0.5. Subsequent queries drop disjuncts successively starting with the lowest *idf* term until all the terms have been dropped. After that, the highest *idf* term is dropped, and the generator starts a fresh cycle of successively dropping the lowest *idf* terms.

Our document retriever is given a target hit list size, and successively executes queries from the query generator until the target number of documents has been found. This insures that downstream modules will always be given a consistently sized set of documents to process.

2.1.2 Method 2

Our second query generation algorithm takes advantage of named-entity recognition technology and other lexical resources to chunk natural language questions so that query terms are not broken across constituent boundaries. To identify relevant entities, we use Sepia (Marton, 2003), an information extraction system based on Combinatory Categorial Grammar (CCG). In particular, personal names are recognized so that inappropriate queries are not generated, e.g., a name such as "John Fitzgerald Kennedy" can give rise to legitimate queries involving "John F. Kennedy", "John Kennedy", and "Kennedy", but never "John Fitzgerald" or simply "John". For certain classes of named-entity types, we have encoded a set of heuristic rules that generates the acceptable variants. Our query generator takes advantage of Lucene's ability to execute phrase queries to ensure that the best matching documents are returned.

Our second query generator also leverages WordNet to identify multi-word expressions that should not be separated in the query process. Multi-token collocations such as "hot dog" should never be broken down into `hot and dog`, since the two interpretations have dramatically different meanings. Because these multi-word expressions can not be predicted by syntax (e.g., compare "hot dog" with "fast car") one practical solution is to employ a fixed list of such lexical items. If a query term is neither a recognized entity nor a multi-word expression, our second query generator expands the term with inflectional and derivational variants using the same technique as the first method.

We discovered that our first query generation method traded off precision for recall with its elaborate term dropping strategy. The result is often a hit

[1] `jakarta.apache.org/lucene/docs/index.html`

list that has simply been "padded" with irrelevant documents; loose queries with few terms are simply not precise enough to retrieve good candidate documents. As an alternative, we implemented a slightly different strategy for our second query generator. It simply drops query terms in order of increasing *idf* until no terms are left and then stops. As a simple example, if a query has three (non-stopword) terms, A, B, and C, arranged in increasing *idf*, our second query generator would produce the following queries:

$$e(A) \wedge e(B) \wedge e(C)$$
$$e(B) \wedge e(C)$$
$$e(C)$$

where $e(x)$ represents the expansions of individual query terms described above.

2.2 Passage Retrieval

The next stage in the processing pipeline is passage retrieval, which attempts to narrow down the set of candidate documents into a set of candidate passages, which are sentences in our architecture.

In a separate study of passage retrieval algorithms (Tellex et al., 2003), we determined that IBM's passage scoring method (Ittycheriah et al., 2000; Ittycheriah et al., 2001) produced the most accurate results. To determine the best passage (sentence in our case), our system breaks each candidate document into sentences and scores each one based on the IBM algorithm.

The IBM passage retrieval algorithm computes a series of distance measures for each passage. The "matching words measure" sums the *idf* values of words that appear in both the query and the passage. The "thesaurus match measure" sums the *idf* values of words in the query whose WordNet synonyms appear in the passage. The "mis-match words measure" sums the *idf* values of words that appear in the query and not in the passage. The "dispersion measure" counts the number of words in the passage between matching query terms, and the "cluster words measure" counts the number of words that occur adjacently in both the question and the passage. These various measures are linearly combined to give the final score for a passage.

We modified the IBM passage scoring algorithm to take into account linguistic knowledge provided by our query generator. The modified algorithm includes scores for matching hyponyms, inflectional variants, derivational variants, and antonyms (negative weight). In addition, our modified algorithm takes advantage of multi-word expressions tokenized from the question, that is, occurrences of "hot" and "dog" within a passage will not match "hot dog".

One of our goals is to determine the effects of additional linguistic knowledge on performance, and for our TREC submissions, we set up a matrix experiment with two query generators and two passage retrievers. The results will be discussed later in Section 5.

2.3 Answer Extraction

The first step of the answer extraction process is to determine the question focus (which was also used in other stages of the processing pipeline); that is, the type of entity asked about in the list question. For this, we enlisted the parser of the START question answering system (Katz, 1997).

Separately, we have compiled offline a large knowledge base of entities, mostly in the form of fixed lists. For example, we have gathered lists of U.S. states, major U.S. cities, major world cities, countries, person names, etc. If the question focus is among one of these types for which we have a fixed list, our answer extractor simply extracts instances of the target type from the top ranking passages collected from the previous stage.

As a simple example, consider the following question:

> In which U.S. states have there been fatalities caused by snow avalanches? (q2183)

Our system correctly identifies the question focus as "U.S. state", and extracts all instances of U.S. states from top ranking passages. Since the passage retrieval algorithm returns passages that already have occurrences of terms from the question, instances of the question focus are likely to be the correct answer.

If the question focus is not in our knowledge base, we employ two backoff procedures. Occasionally, answers to list questions have the question focus directly embedded in them, e.g., "littleneck clam" is a type of clam, and in the absence of any additional knowledge, noun phrases containing the focus are extracted as answer instances. Finally, if no noun phrases containing the question focus can be found, our answer extraction module simply picks the noun phrase closest to the question focus in each of the passages.

After collecting all the answer candidates, we discard ones that have query terms in them. Noun phrases containing keywords from the query typically repeat some aspect of the query and make little sense as answers. This heuristic worked well in our previous question answering system (Lin et al., 2002).

2.4 Duplicate Removal

Answer instances extracted from the previous stage typically contain duplicates, which our system removes using a thresholded edit-distance measure. Finally, our system computes the number of answer instances to return based on a relative thresholding scheme. Each answer candidate is given a score equal to the score of the passage from which it was

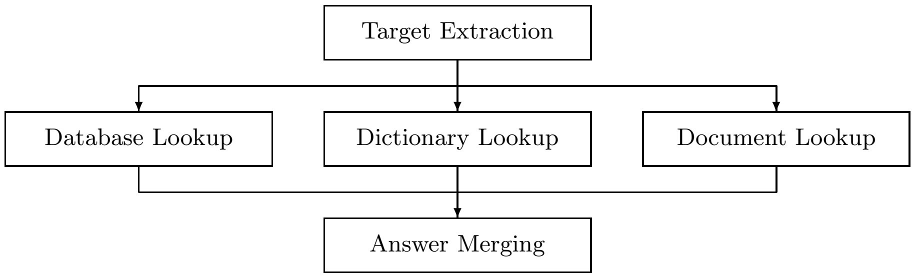

Figure 2: Architecture for answering definition questions.

extracted, and all candidate answers below 10% of the maximum score are discarded. The remaining instances are returned as the final answer.

3 Definition Questions

Our architecture for answering definition questions is shown in Figure 2. The target extraction module first analyzes the natural language question to determine the unknown term. Once the target term has been found, three parallel techniques are employed to retrieve relevant nuggets that define the term: lookup in a terminological database created from the AQUAINT corpus, lookup in a Web dictionary followed by answer projection, and lookup directly in the AQUAINT corpus with information retrieval techniques. Answers from the three different sources are merged to produce the final system output. The following subsections will describe each of these techniques in greater detail.

3.1 Target Extraction

We developed a pattern-based parser to analyze definition questions and extract the target term to be defined using simple regular expressions.

If the natural language question does not fit any of our patterns, the parser heuristically extracts the last sequence of capitalized words in the question as the target. Our simple definition target extractor was tested on definition-style questions from the previous TREC evaluations and performed quite well on those training questions.

3.2 Database Lookup

The use of surface patterns for answer extraction has proven to be an effective strategy for question answering. Typically, surface patterns are applied to a candidate set of documents that have been returned by traditional document retrieval systems. While this strategy may be effective for factoid questions, it generally suffers from low recall. In the case of factoid questions, where only one instance of an answer is necessary, recall is not a primary concern. However, definition questions require a system to find as many relevant nuggets as possible, making recall very important.

Instead, we employ an alternative strategy: by applying a set of surface patterns offline, we are able to "precompile" from the AQUAINT corpus knowledge nuggets about every entity mentioned within it. In essence, we have automatically constructed an immense knowledge base containing nuggets distilled from every article within the corpus. The task of answering definition questions becomes a simple database lookup.

Our surface patterns operate both at the word level and part-of-speech level. We utilize patterns over part-of-speech tags to perform rudimentary chunking, i.e., marking the boundaries of noun phrases. The surface patterns we used include the following (Figure 3 shows several examples):

- **Copular pattern.** Copular constructions often provide a definition of the target term. In order to filter out spurious nuggets (e.g., progressive tense), our system throws out all definitional nuggets that do not begin with a determiner; this ensured that we only get "NP_1 be NP_2" patterns, where NP_1 is usually the target term, and NP_2 the nugget.
- **Appositive pattern.** Commas typically provide strong evidence for the presence of an appositive. With the assistance of part-of-speech tags, identifying "NP_1, NP_2" patterns is relatively straightforward. Most often, NP_1 is the target term and NP_2 is the nugget, but occasionally the positions are swapped. Thus, we index both NPs as the target term.
- **Occupation pattern.** Common nouns preceding proper nouns typically provide some relevant information such as occupation, affiliation, etc. In order to boost the precision of this

copular pattern: A **fractal** *is* a pattern that is irregular, but self-similar at all size scales
appositive pattern: The **Aga Khan**, Spiritual Leader of the Ismaili Muslims
occupation pattern: steel magnate **Andrew Carnegie**
verb pattern: **Althea Gibson** *became* the first black tennis player to win a Wimbledon singles title
parenthesis pattern: **Alice Rivlin** *(*director of the Office of Management and Budget*)*

Figure 3: Sample nuggets extracted from the AQUAINT corpus using surface patterns. The target terms are in bold, the nuggets underlined, and the pattern landmarks in italics.

pattern, our system discards all common noun phrases that do not contain an "occupation", e.g., actor, spokesman, leader, etc. We mined the list of occupations from WordNet and Web resources.

- **Verb pattern.** By statistically analyzing a corpus of biographies of famous people, we were able to compile a list of verbs that are commonly used to describe people and their accomplishments, e.g., *become*, *founded*, *invented*, etc. This list of verbs is employed to extract "NP_1 *verb* NP_2" patterns, which are usually very precise.
- **Parenthesis pattern.** Parenthetical expressions following noun phrases typically provide some interesting nuggets about the preceding noun phrase; for persons, it often contains birth/death years or occupation/affiliation.

Typically, our patterns identify short nuggets on the order of a few dozen characters. In answering definition questions, we decided to return responses that include additional context. To accomplish this, we simply expand all nuggets around their center point to encompass one hundred characters. We found that this technique enhances the readability of our responses: many nuggets seem odd and out of place without context and surrounding text is often necessary for disambiguation. Furthermore, returning a longer answer means that our responses sometimes contain additional relevant nuggets that are not part of the original pattern. In definitions questions, these "idiosyncratic" nuggets occur quite often.

One drawback to our knowledge base of nuggets is a tremendous amount of redundancy. Because we compiled all patterns from all entities within the AQUAINT corpus, common nuggets are often repeated. In order to deal with this, we developed a simple algorithm to remove duplicate information: if any two responses share more than sixty percent of their keywords, one of them is thrown out.

3.3 Dictionary Lookup

Another component of our system for answering definition questions utilizes an existing Web-based dictionary for nuggets. The motivation is simple: extensive resources already exist on the Web that can supply a wealth of nuggets for defining terms. Why not take advantage of these sources? Obviously, such an approach cannot be applied directly, because all nuggets must originate from the AQUAINT corpus. To address this issue, we developed answer projection techniques to "map" dictionary definitions back onto AQUAINT documents. The mapping component is based on the insight that if you already know the answer, it is much easier to find relevant nuggets in the corpus.

Given the target term, our dictionary wrapper goes online to the Merriam-Webster website and fetches the term's definition. Keywords from the definition are used as the input query to the Lucene document retriever. Once a set of candidate documents has been returned, we chunk all the sentences and score each one based on its keyword overlap with the dictionary definition. The sentences with the highest scores are retained and shortened to one hundred characters.

3.4 Document Lookup

As a last resort, our system employs simple document retrieval to extract relevant nuggets if no answers are found with the first two techniques. The target term is used as a query to Lucene to gather a set of candidate documents. These documents are chunked into separate sentences and sentences containing the target terms are retained as responses.

3.5 Answer Merging

The input to the answer merging stage is a series of one hundred character responses from each of the sources: database, dictionary, and documents. The responses are arranged according to an ad-hoc priority scale we developed based on the accuracy of each approach. For example, we found that verb patterns generally returned very good nuggets, and copular constructions were often less accurate. The priority of dictionary answers is somewhere between the best and worst patterns, ordered such that some dictionary responses (if any) would always be returned in the final answer. Responses extracted directly from document lookup are used only if the two other methods return no answers: document lookup is considered a strict back-off method used only as a last resort.

Finally, the answer merging stage of our system also decides the number of one hundred character responses to return. Since the length penalties for returning long answers are not very steep, we decided to return longer answers in hope of encompassing more relevant nuggets. Given n responses, we calculated the final number of responses to return as:

$$\begin{array}{ll} n & \text{if } n \leq 10 \\ n + \sqrt{n-10} & \text{if } n > 10 \end{array}$$

This ensures that our system will always return a generous number of nuggets most of the time.

4 Factoid Questions

Our system for answering factoid questions was largely unchanged from last year. We employed the Aranea question answering system (Lin et al., 2002; Lin and Katz, 2003), which embraces two different views of the World Wide Web: as a heterogeneous collection of unorganized documents and as a source of carefully crafted and organized knowledge about specific topics.

Aranea's approach is primarily motivated by an observation that the distribution of user queries quantitatively obeys Zipf's Law—a small fraction of question types accounts for a significant portion of all question instances. Large classes of commonly-occurring questions translate naturally into database queries and are handled by Aranea using a technique we call *knowledge annotation*, which allows our system to access semistructured and heterogeneous data as if it were a uniform database. After identifying useful resources on the Web, we manually craft site-specific wrappers that provide uniform access capabilities to the knowledge contained in those resources under an object–property–value data model (Katz et al., 2002). Natural language questions are translated into database queries through schemata, which are then executed to provide the final answer.

As with all Zipf curves, there is a broad tail where individual instances are either unique or account for an insignificant fraction of total questions. To answer questions that cannot be easily classified into common categories or grouped by simple patterns, Aranea employs what we call redundancy-based *knowledge mining* techniques. Knowledge mining leverages the massive amounts of information available on the Web to overcome many thorny problems associated with natural language processing. The insight is simple: the more data available, the greater the chance that the answer to a natural language question is stated simply as a reformulation of that question. In such cases, simple pattern matching techniques suffice to accurately extract answers.

4.1 Answer Projection

The setup of the TREC evaluation required supporting documents for each answer. Since Aranea does not directly use the AQUAINT corpus in the question answering process, Web-based answers must then be projected back onto AQUAINT documents. 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. For the set of candidate documents, we tried three different approaches: using the NIST-supplied PRISE documents, using documents generated by our first query generation algorithm (Section 2.1.1), and using documents generated by our second query generation algorithm (Section 2.1.2).

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 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.

5 Results

A summary of our results at this year's TREC evaluation is shown in Table 1. For list questions, the query generator and passage scoring algorithms used for each of the runs is shown in Table 2. For definition questions, all three submissions were exactly the same. For factoid questions, the query generation algorithm used for answer projection in each of the runs is shown in Table 2.

5.1 List Results

In general, the modified IBM passage scoring algorithm performed slightly worse than the original IBM algorithm. However, for the most part, they returned exactly the same responses; it is difficult to determine if the score differences are above the margin of error inherent in human judgments. In retrospect, we believe that our modified IBM algorithm was too lax in matching various forms of expansions (too high a score was given to variants). It is a well-known result that uncontrolled expansion of lexical-semantic relations (e.g., synonyms and hyponyms) results in lower performance (Voorhees, 1994). It has likewise been shown that inflectional

Task	MITCSAIL03a	MITCSAIL03b	MITCSAIL03c	best	median	worst
Factoid	0.293	0.295	0.291	0.7	0.177	0.034
List	0.13	0.118	0.134	0.396	0.069	0
Definition	0.309	0.282	0.282	0.555	0.192	0
weighted total	0.256	0.248	0.250			

Table 1: Summary of MIT CSAIL submissions.

	MITCSAIL03a	MITCSAIL03b	MITCSAIL03c
List questions:			
Query generator	method 1	method 2	method 2
Passage retriever	IBM	IBM	modified IBM
Factoid questions:			
Answer projection	PRISE	method 1	method 2

Table 2: Variations in each of the TREC runs.

and derivational expansion does not increase performance. However, these experiments were conducted under different circumstances for a traditional document retrieval task, which is significantly different from the task of extracting succinct answers. For the question answering task, we believe that linguistically-motivated query expansions will have a positive impact on performance. While our experiments have not yet shown a significant overall positive effect, we attribute this to implementational deficiencies in our overall system, rather than conceptual shortcomings.

Our second query generation method performed slightly better than our first query generation method. In particular, tokenization of multi-token expressions had the most positive impact on performance. Consider the following question:

> What countries have had school bus accidents that resulted in fatalities? (q2180)

The second query generation algorithm correctly identified "school bus" as a collocation and thus never broken up the expression into "school" and "bus".

5.1.1 Question Focus

Our strategy for answering list questions depends on correctly identifying the question focus, i.e., the type of entity sought after. For a few questions, our system was unable to correctly determine the question focus, resulting in a score of zero for those questions. To address this shortcoming, we will improve START's ability to recognize question focus.

Although knowledge of the question focus helps in answering a question, care is needed to map the focus word into a corresponding class of entities. Consider the following questions:

> List the names of cell phone manufacturers. (q2096)
> Name recipients of funds given by the various foundations of Bill and Melinda Gates. (q2291)

Our system correctly identified "manufacturer" as the question focus in the first question, but chose the wrong sense. The term was on our list of professions, so the system incorrectly looked for personal names. The second question demonstrates that not all targets, even when correctly identified, are useful. "Recipients" are so general that they can be anything: people, companies, organizations, even countries.

Not surprisingly, our system performed well for questions whose foci had corresponding fixed lists in our knowledge base. Since we had exhaustive lists for entities like cities, countries, and U.S. presidents, all our answers were at least of the correct type. However, since our system ignored syntactic relations within the passage, it often overgenerated irrelevant answers. Consider the following question:

> What countries have won the men's World Cup for soccer? (q2346)

Since our system returned all countries found near the relevant keywords, most of the answers were countries that played in the World Cup, not winners of it. As a result, we obtained high recall, but poor precision, on this question. This is certainly a case where the use of syntactic relations can dramatically improve question answering performance (Katz and Lin, 2003).

Our backoff method of looking for the question focus in candidate answers worked for the following question:

	MITCSAIL03a		MITCSAIL03b		MITCSAIL03c	
	PRISE		Method 1		Method 2	
Right	121	29.30%	122	29.54%	120	29.06%
Inexact	18	4.36%	15	3.63%	15	3.63%
Unsupported	26	6.30%	21	5.08%	21	5.80%
Wrong	248	60.05%	255	61.74%	257	62.23%
Total	413		413		413	

Table 3: Detailed analysis of factoid questions

What grapes are used in making wine? (q1940)

The system extracted correct answers like "Chardonnay Grapes". However, the same technique didn't work when the question focus was "team" or "food" because journalists typically do not write "X team" or "Y food".

5.2 Definition Results

Although the responses were identical in each of our three submitted runs for definition questions, the scores were not; that is, given the same exact answer string, assessors came up with different judgments some of the time. This can be attributed to the margin of error inherent in human judgments. Out of the 317 responses we submitted for the 50 definition questions, there were 19 responses which were not judged the same over all three runs. However, 7 of these were cases where assessors found the same nugget in different responses for a question. In addition, there are clear instances where an answer nugget is in one of our responses and the assessors missed it, even when the nugget was present word for word. We suspect that our decision to return one hundred character responses contributed to these variations in judgment.

Target extraction was the single biggest source of error in answering definition questions. If the target term is not correctly identified, then all subsequent modules have little chance of providing relevant nuggets.

We did not anticipate the presence of stopwords in names. Consider the following questions:

What is Bausch & Lomb? (q1917)
Who is Vlad the Impaler? (q1933)
Who is Akbar the Great? (q1955)

Our naive pattern based parser extracted "Lomb", "Impaler" and "Great" as the target terms for the above questions, respectively. Fortunately, "Impaler" is such a rare word that we actually returned nuggets concerning "Vlad the Impaler". Similarly, "Lomb" so frequently co-occurs with "Bausch & Lomb" that our system was able to provide relevant nuggets. However, since "Great" is a very common word, our definitions for "Akbar the Great" were mostly meaningless.

Our inability to parse certain forms of names is related to our simple assumption that the final consecutive sequence of capitalized words in a question is the target. This simply turned out to be an incorrect assumption:

Who was Abraham in the Old Testament? (q1972)
What is ETA in Spain? (q1987)
What is Friends of the Earth? (q2222)

Our pattern-based parser extracted "Old Testament", "Spain", and "Earth" as the targets for those questions, respectively. The inability to correctly identify the target term directly resulted in our failure to return relevant nuggets.

Another problem our target extractor encountered is apposition. Take the following example:

What is the medical condition shingles? (q2348)

Our target extractor incorrectly identified "medical condition shingles" as the target term. As a result, our system did not identify a single relevant nugget. To better extract target terms for definition questions, we will employ START and Sepia in the future, which we were unable to utilize for definition questions this year for technical reasons.

5.3 Factoid Results

Table 3 shows a detailed analysis of factoid questions. As in previous years, answer projection appears to be the biggest Achilles' heel in our Web-based question answering strategy, as shown by the relatively large fraction of unsupported and inexact answers (in comparison to typical results of other teams). Furthermore, it does not appear that any of our more advanced query generation algorithms had any significant impact of the final score of factoid questions.

6 Conclusion

The focus of our research this year was to integrate Web- and corpus-based question answering techniques under a unified framework. This falls under our general research agenda of employing linguistic techniques, at the lexical, morphological, syntactic, and semantic levels in conjunction with statistical techniques when appropriate. Although our TREC experiments have yet to show significant benefits from linguistically-motivated processing techniques, our research has demonstrated the effectiveness of linguistically sophisticated techniques for question answering within more restricted domains. We believe that high performance in the question answering task can only be achieved through fusion of multiple strategies and multiple resources.

References

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)*.

Charles Clarke, Gordon Cormack, Thomas Lynam, C.M. Li, and Greg McLearn. 2001. Web reinforced question answering (MultiText experiments for TREC 2001). In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2002)*.

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. 2000. IBM's statistical question answering system. In *Proceedings of the Eighth Text REtrieval Conference (TREC-9)*.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system—TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*.

Boris Katz and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*.

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. 2002. 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. 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)*.

Jimmy Lin and Boris Katz. 2003. Question answering from the Web using knowledge annotation and knowledge mining techniques. In *Proceedings of Twelfth International Conference on Information and Knowledge Management (CIKM-2003)*.

Jimmy Lin, Aaron Fernandes, Boris Katz, Gregory Marton, and Stefanie Tellex. 2002. Extracting answers from the Web using knowledge annotation and knowledge mining techniques. In *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*.

Gregory A. Marton. 2003. Sepia: Semantic parsing for named entities. Master's thesis, Massachusetts Institute of Technology.

Stefanie Tellex, Boris Katz, Jimmy Lin, Gregory Marton, and Aaron Fernandes. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2003)*.

Ellen M. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-1994)*.