A vast amount of information is available on the Internet, and many tools have been developed to gather it. These include search engines such as AltaVista, Infoseek, Lycos, and many others. A main problem with current search engines is that broad, general queries produce a large volume of documents, many of which are totally irrelevant. At the same time, many relevant documents can be missing because the query does not contain the keywords that index them; for the same reason, specific queries often fail to produce any documents at all. Boolean operators can sometimes help, but they can also further restrict a query such that it fails to find relevant documents.

The lack of a natural language interface is another limitation of current search engines. Many users, particularly those who are not computer professionals, would prefer to ask, “Who were the U.S. presidents of the past century?” rather than form a Boolean query such as (US NEAR presidents) AND (past NEAR century). These users would undoubtedly benefit from an interface that could transform sentences into Boolean queries. But there is another, perhaps even greater advantage in using natural language questions. With a modest amount of linguistic processing, the words in an English question can be “disambiguated” and the query subsequently expanded to include similar words from online dictionaries.

In this article, we describe such a system for broadening Web searches. The large number of documents that result from the search are then subjected to a new search using an operator that further capitalizes on natural language constructs by extracting only the paragraphs that render information relevant to the query. We conclude with test results that show significant improvements in two metrics:

- **Performance** is a standard information-retrieval system measure of the number of relevant documents retrieved over the total number of documents retrieved;
- **System productivity** is the percentage of questions answered satisfactorily, a new measure that we introduce to address the Internet environment.
INTERFACE SYSTEM ARCHITECTURE

Figure 1 shows the system architecture. The input query or sentence expressed in English is first presented to the lexical processing module. This module was adopted from an information extraction system that we developed for the Message Understanding Conference (MUC) competition. The word and sentence boundaries are located via a process called tokanization. The words are tagged for their part of speech using a version of Brill’s tagger. A phrase parser segments each sentence into constituent noun and verb phrases and recognizes the head words. After eliminating stopwords (conjunctions, prepositions, pronouns, and modal verbs), we are left with some keywords that represent the important concepts of the input sentence.

In the next three sections, we describe the word-sense-disambiguation (WSD), query-expansion, and postprocessing modules in our system. The current implementation uses WordNet for WSD and query expansion, and the AltaVista search engine for Internet search. For more information on these tools, see the sidebar “Development Resources for Improving Internet Searches.”

WORD-SENSE DISAMBIGUATION

Word-sense disambiguation is a novelty of our system. Each keyword in the query is mapped into its corresponding semantic form as defined in WordNet. This step enables subsequent query expansion based on semantic concepts rather than keywords.

Our approach takes advantage of the sentence context. The words are paired, and each word is disambiguated by searching the Internet with queries formed using different senses of one word while keeping the other word fixed. The senses are ranked simply by the number of hits. In this way all the words are processed and senses are ranked.

The next step refines the ordering of senses by using a semantic density method that measures the number of common words within a semantic distance of two or more words. The method uses WordNet glosses. The algorithms and performance results are presented in the remainder of this section (for an example application of the algorithms, see the sidebar “Applying the WSD Algorithms”).

Algorithm 1: Contextual Ranking of Word Senses

From a semantically untagged word pair \((W_1, W_2)\), we first select one of the words, say \(W_2\), and form a similarity list for each of its senses, using WordNet’s synset for that word.

Consider, for example, that \(W_2\) has \(m\) senses. This means that \(W_2\) appears in \(m\) similarity lists:

\[
\begin{align*}
(W_1, W_2^1, W_2^2, \ldots, W_2^{k_1}) \\
(W_1, W_2^1, W_2^2, \ldots, W_2^{k_2}) \\
\vdots \\
(W_1, W_2^1, W_2^2, \ldots, W_2^{k_m})
\end{align*}
\]

where \(W_1, W_2^1, \ldots, W_2^m\) are the senses of \(W_2\), and \(W_2^{j_0}\) represents the synonym number of the sense \(W_2^j\) as defined in WordNet. We can then form \(W_1 - W_2^{j_0}\) pairs, specifically:

\[
\begin{align*}
(W_1, W_2^1, W_1 - W_2^{j_1}, \ldots, W_1 - W_2^{k_1}) \\
(W_1, W_2^1, W_1 - W_2^{j_2}, \ldots, W_1 - W_2^{k_2}) \\
\vdots \\
(W_1, W_2^1, W_1 - W_2^{j_m}, \ldots, W_1 - W_2^{k_m})
\end{align*}
\]
Finally, we perform an Internet search for each set of \(W_1 - W_2^{(i)}\) pairs. The query uses the operators provided by AltaVista to find occurrences of \(W_1\) together with that sense of \(W_2\) for each set. For example, one such query is

\[
(W_1*W_2^{(1)}* OR W_1*W_2^{(1)*} OR W_1*W_2^{(2)*} OR \ldots OR W_1*W_2^{(m)*}) \text{ for all } 1 \leq i \leq m.
\]

The asterisk (*) is used as a wild card to increase the number of hits with morphologically related words. Using such a query, we get the number of hits for each sense \(i\) of \(W_2\), and this provides a ranking of the \(m\) senses of \(W_2\) as they relate with \(W_1\).

A similar algorithm is used to rank the senses of \(W_1\) while keeping \(W_2\) constant (un-disambiguated). Since these two procedures are performed over a large corpora (the Internet) and with the help of similarity lists, there is little correlation between the results they produce.

Evaluation of Algorithm 1. We tested this method on 384 pairs: 200 verb-noun, 127 adjective-noun, and 57 adverb-verb extracted from the first text of the SemCor 1.6 from the Brown corpus. Using the AltaVista query form, we obtained the results shown in Table 1 (on page 38).

The table indicates the percentages of correct senses (as given by SemCor) ranked by us as the first, second, third, and fourth choices of our list.
We concluded that keeping the top four choices for verbs and nouns and the top two choices for adjectives and adverbs would cover all relevant senses in the mid and upper 90 percent range.

From one point of view, a possible use of the procedure so far is to exclude senses that do not apply. This can save considerable computation time as many words are highly polysemous.

Algorithm 2: Conceptual Density Ranking
A measure of the relatedness between words can be a knowledge source for several decisions in natural language processing (NLP) applications. Our approach is to construct a linguistic context for each sense of the verb and noun, and to measure the number of nouns shared by the verb and the noun contexts.

In WordNet each concept has a gloss that acts as a microcontext for that concept. This rich source of linguistic information proved useful in determining the conceptual density between words, though it applies only to verb-noun pairs and not to adjectives or adverbs.

We developed an algorithm that takes a semantically untagged verb-noun pair and a ranking of noun senses (as determined by Algorithm 1) as its input and gives a sense-tagged verb-noun pair as output. Given a verb-noun pair V - N, we use WordNet to determine the possible senses of the verb and the noun, <v1, v2, ..., vn1> and <n1, n2, ..., nl>, respectively. Then we use Algorithm 1 to rank the senses of the noun. Only the first t possible senses of this ranking will be considered; the rest are dropped to reduce the computational complexity.

By setting the threshold t = 2, we kept only senses #2 and #3. (The notation #i/n means sense i out of n possible senses given by WordNet et.)

Next, we applied Algorithm 2 to rank the four possible combinations (two for the verb times two for the noun). Table A summarizes the results, according to Equation 1 from the main text.

The largest conceptual density, C12 = 0.30, corresponds to v1 - n2: revise#1/2 - law#2/5. This combination of verb-noun senses also appears in SemCor, file br-a01.

Applying the WSD Algorithms

Consider the verb-noun collocation revise law. The verb revise has two possible senses in WordNet 1.6, and the noun law has seven senses.

We applied Algorithm 1 and searched the Internet using Alta Vista for all possible pairs V - N that can be created using revise and the words from the similarity lists of law. We obtained the following ranking of senses: law#2(2,829), law#3(648), law#4(640), law#6(397), law#1(224), law#5(37), law#7(10), where the number in the parentheses indicates the number of hits.

By setting the threshold t = 2, we kept only senses #2 and #3. (The notation #i/n means sense i out of n possible senses given by WordNet et.)

Next, we applied Algorithm 2 to rank the four possible combinations (two for the verb times two for the noun). Table A summarizes the results, according to Equation 1 from the main text.

The largest conceptual density, C12 = 0.30, corresponds to v1 - n2: revise#1/2 - law#2/5. This combination of verb-noun senses also appears in SemCor, file br-a01.

Table A. Values used in computing the conceptual density Cij.

| | \(|cdij|\) | descj | Cij |
|---|---|---|---|
| v1 | n2 | n3 | 975 | 1,265 | 0.30 | 0.28 |
| v2 | 0 | 0 | 975 | 1,265 | 0 | 0 |

| | \(|cdij|\) = Number of common concepts between verb and noun hierarchies. |
| | descj = Number of nouns within the hierarchy of each sense nj. |
| | \(Cij\) = Conceptual density for each pair vi - nj. |

We concluded that keeping the top four choices for verbs and nouns and the top two choices for adjectives and adverbs would cover all relevant senses in the mid and upper 90 percent range.

From one point of view, a possible use of the procedure so far is to exclude senses that do not apply. This can save considerable computation time as many words are highly polysemous.
and descendents\(_j\) is the total number of words within the hierarchy of noun \(n_i\).

Given the conceptual density \(C_{ij}\), the last step of Algorithm 2 ranks each pair \(v_i - n_j\), for all \(i\) and \(j\).

**Rationale for Algorithm 2.** This algorithm capitalizes on WordNet's gloss, which explains a concept and provides one or more examples with typical usage of that concept. To determine the most appropriate noun and verb hierarchies, we performed some experiments using SemCor and concluded that the noun subhierarchy should include all the nouns in the class of \(n_j\). The subhierarchy of verb \(v_i\) is taken as the hierarchy of the highest hypernym \(h_i\) of the verb \(v_i\). It is necessary to consider a larger hierarchy than just the one provided by synonyms and direct hyponyms. As we replaced the role of a corpora with glosses, we achieved better results with more glosses. Still, we don’t want to enlarge the context too much.

The nouns with a big hierarchy tend to have a larger value for \(|c_{ij}|\), so the weighted sum of common concepts is normalized in respect to the dimension of the noun hierarchy. Since a hierarchy’s size grows exponentially with its depth, we used the logarithm of the total number of descendents in the hierarchy, \(\log(\text{descendents}_j)\). We experimented with a few other metrics, but after running the program on several examples, the formula from Algorithm 2 provided the best results.

**Evaluation of WSD Method**

Table 2 shows the overall results using Algorithm 1 followed by Algorithm 2 on 384 word pairs. Comparing Table 2 results with those for Table 1 will show the percentage increase in accuracy contributed by Algorithm 2 beyond Algorithm 1.

To our knowledge, there is only one other method, recently reported, that disambiguates unrestricted nouns, verbs, adverbs, and adjectives in texts.\(^3\) The method uses WordNet and attempts to exploit sentential and discourse contexts; it is based on the idea of semantic distance between words and on lexical relations. There are several accurate statistical methods, such as the one presented in Yarowsky,\(^4\) but they disambiguate only one part of speech (nouns in this case) and focus on only a few words because they lack training corpora.

Table 3 presents a comparison between our results and the results reported in those papers. The baseline for the comparison is the occurrences of the first senses from WordNet. For applications such as query expansion in information retrieval, our method has the additional advantage of potentially considering the first two senses for each word, in which case the average accuracy (as determined in Table 2) is 91 percent.

### QUERY EXPANSION

The technology of query expansion is almost 30 years old.\(^5\) It can be used either to broaden the set of documents retrieved or to increase the retrieval precision. In the former case, the query is expanded with terms similar to those from the original query, while in the second case, the expansion procedure adds completely new terms. We take the first approach, using WordNet to find words semantically related to concepts from the original query. (An example of the second technique is the Smart system, developed at Cornell University, which uses words derived from documents relevant to the original query.\(^6\))

The query expansion module in our system has two main functions:

- the construction of similarity lists using WordNet, and
- the formation of the actual query.

Once we have a sense ranking for each word of the input sentence, it is relatively easy to use WordNet's rich semantic information to identify many words that are semantically similar to a given input word. Doing this increases the chance of finding more answers to input queries. WordNet can provide semantic similarity between words that belong to the same synonym set.
Consider, for example, the word activity. WordNet gives seven senses for this word. The synset for the first sense includes two other synonyms, action and activeness. The similarity list for this sense of the word is therefore

\[ W = \{ \text{action, activity, activeness} \} \]

The efficacy of expanding a query for search in large text collections was investigated by Voorhees. She used WordNet to experiment with four expanding strategies:

- by synonyms only,
- by synonyms plus all descendants in a isa hierarchy,
- by synonyms plus parents and all descendants in a isa hierarchy, and
- by synonyms plus any synset directly related to the given synset.

Her results showed no significant differences in the precision obtained using any one of these four expanding strategies.

Let's denote with \( x_i \) the words of a question or sentence, and with \( W_i = \{ x_k, x_l \} \) the similarity lists provided by WordNet for each word \( x_i \). The elements of a list are \( x_k \) where \( k \) enumerates the elements in each list (that is, the words on the same level of similarity with the word \( x_i \)). We can now use these lists to formulate the actual query, using the Boolean operators accepted by current search engines. The OR operator is used to link words within a similarity list \( W_i \), while the AND and NEAR operators link the similarity lists.

While different combinations of similarity lists linked by AND or NEAR operators are possible, two basic forms

\[ W_1 \text{ AND } W_2 \text{ AND } ... \text{ AND } W_n \]
\[ W_1 \text{ NEAR } W_2 \text{ NEAR } ... \text{ NEAR } W_n \]

give the maximum and minimum, respectively, of the number of documents retrieved. In most cases, the maximum format gathered thousands of documents, while the minimum format almost always had null results.

We can assume that any documents containing the answers must be among the large number of documents provided by the AND operators, but the search engine failed to rank them in the top of the list. Thus, we sought a new operator that would filter out many irrelevant texts.

### POSTPROCESSING WITH A NEW OPERATOR

Our approach to filtering documents is to first search the Internet using weak operators (AND, OR) and then to further search this large number of documents using a more restrictive operator. For this second phase, we propose the following additional operator:

\[ \text{PARAGRAPH} n (\ldots \text{similarity lists} \ldots ) \]

The PARAGRAPH operator searches as an AND operator for the words in the similarity lists, but with the constraint that the words belong only to some \( n \) consecutive paragraphs, where \( n \) is a positive integer. The parameter \( n \) selects the number of paragraphs, thus controlling the size of the text retrieved from a document considered relevant. The rationale is that most likely the information requested is found in a few paragraphs rather than being dispersed over an entire document. (A similar idea can be found in Callan.)

To apply this new operator, the documents gathered from the Internet must be segmented into sentences and paragraphs. Separating a text into sentences is an easy task that can be solved just by using the punctuation. However, the unstructured texts on the Web make paragraph segmentation much more difficult. Both Callan and Hearst have developed work in this direction, but their methods work only for structured texts containing lexical separators known a priori (for example, a tag or an empty line). Thus, we had to use a method that covers almost all possible paragraph separators that can occur in Web texts. The paragraph separators that we've considered so far are HTML tags, empty lines, and paragraph indentations.

We give a complete example of our system in the sidebar, “Finding a Relevant Answer: A Query Example.”

### TEST RESULTS

To test our system, we used 50 questions from real Internet searches and 50 questions derived from

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Stetina</th>
<th>Yarowsky</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun</td>
<td>80.3%</td>
<td>85.7%</td>
<td>93.9%</td>
<td>86.5%</td>
</tr>
<tr>
<td>Verb</td>
<td>62.5%</td>
<td>63.9%</td>
<td>67%</td>
<td>79.8%</td>
</tr>
<tr>
<td>Adjective</td>
<td>81.8%</td>
<td>83.6%</td>
<td>79.8%</td>
<td>87%</td>
</tr>
<tr>
<td>Adverb</td>
<td>84.3%</td>
<td>86.5%</td>
<td>87%</td>
<td>80.1%</td>
</tr>
<tr>
<td>Average</td>
<td>77%</td>
<td>80%</td>
<td>80.1%</td>
<td>80%</td>
</tr>
</tbody>
</table>
Finding a Relevant Answer: A Query Example

Suppose you want to answer the question: “How much tax does an average salary worker pay in the United States?”

The linguistic processing module (shown in Figure 1, main text) identified keywords, including part-of-speech tags, which were then ranked for word sense as follows:

\[
\begin{align*}
& x_1 = \text{(tax), pos = noun, sense #1/1} \\
& x_2 = \text{(average), pos = adjective, sense #4/5} \\
& x_3 = \text{(salary), pos = noun, sense #1/1} \\
& x_4 = \text{(the United States), pos = noun, sense #1/2} \\
& x_5 = \text{(worker), pos = noun, sense #1/4} \\
& x_6 = \text{(pays), pos = verb, sense #1/7}
\end{align*}
\]

The sense number indicates the actual WordNet sense that resulted from the disambiguation of all possible senses in WordNet. For instance, adjective average has five senses and the system picked sense #4.

These keywords are the input for the next step of our system. Using the similarity relation encoded in the WordNet synsets, it yields the following six similarity lists:

\[
\begin{align*}
& W_1 = \{\text{tax, taxation, revenue enhancement}\} \\
& W_2 = \{\text{average, intermediate, medium, middle}\} \\
& W_3 = \{\text{salary, wage, pay, earnings, remuneration}\} \\
& W_4 = \{\text{United States, United States of America, America, US, U.S., USA, U.S.A.}\} \\
& W_5 = \{\text{worker}\} \\
& W_6 = \{\text{pay}\}
\end{align*}
\]

These lists are used to formulate queries for the search engine. Table A shows some queries and the number of documents retrieved by AltaVista. Though AltaVista has one of the most powerful sets of operators available from search engines today, the ranking provided by AltaVista is of no use for us here. None of the 10 leading documents in any category provided the desired information. Nor did the single document fetched by Query 4:

"...The proposed tax cut, and the bigger one promised for next year, if enacted, will be paid for by the Social Security wage taxes of middle and low-income workers of America. Employees have been willing to pay these taxes because of the promise of guaranteed Social Security retirement benefits. This Republican tax bill is a betrayal of the low and middle-income workers. The unfairness of these proposals is breathtaking."

Analysis of the table results indicates a gap in the volume of documents retrieved with the AltaVista operators. For instance, using only the AND operator (Query 1) obtained 49,182 documents, but the NEAR operator (Queries 4 and 6) produced only one (irrelevant) and zero outputs, respectively. This operator seems to be too restrictive, while it fails to identify the right answer. Various combinations of AND and NEAR operators achieved no great results.

Using the PARAGRAPH operator, however, the system found a relevant answer:

In 1910, American workers paid no income tax. In 1995, a worker earning an average wage of $26,000 pays about 24% (about $6,000) in income taxes. The average American worker’s pay has risen greatly since 1910. Then, the average worker earned about $600 per year. Today, the figure is $26,000.

<table>
<thead>
<tr>
<th>Query</th>
<th>No. of documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>W_1 \ AND \ W_2 \ AND \ W_3 \ AND \ W_4 \ AND \ W_5 \ AND \ W_6</td>
</tr>
<tr>
<td>2</td>
<td>W_1 \ AND \ (W_2 \ NEAR \ W_3) \ AND \ W_4 \ AND \ W_5 \ AND \ W_6</td>
</tr>
<tr>
<td>3</td>
<td>W_1 \ NEAR \ W_2 \ NEAR \ W_3 \ AND \ W_4 \ AND \ W_5 \ AND \ W_6</td>
</tr>
<tr>
<td>4</td>
<td>W_1 \ NEAR \ W_2 \ NEAR \ W_3 \ AND \ W_4 \ NEAR \ W_5 \ NEAR \ W_6</td>
</tr>
<tr>
<td>5</td>
<td>W_1 \ AND \ {average \ W_3} \ AND \ W_4 \ AND \ W_5 \ AND \ W_6</td>
</tr>
<tr>
<td>6</td>
<td>W_1 \ NEAR \ {average \ W_3} \ AND \ W_4 \ NEAR \ W_5 \ NEAR \ W_6</td>
</tr>
</tbody>
</table>

Table A. Query results with various combinations of operators.

Each of 50 topics defined for ad hoc queries at the Sixth Text Retrieval Conference (TREC-6), cosponsored by the U.S. National Institute of Standards and Technology (NIST) and the Defense Advanced Research Projects Agency (DARPA).

Figure 2 presents an example topic from the TREC-6 ad hoc collection. Each topic is a frame-like data structure with the following fields:

- **<num>** identifies the topic number;
- **<title>** classifies the topic within a domain;
- **<desc>** describes the topic briefly (for TREC-6, this section was intended to be an initial search query);
- **<narr>** provides a further explanation of what a relevant material may look like.

We edited the **<desc>** field to derive natural language questions similar to those normally asked by real users searching the Internet. For example, from the corpus entry presented above, the question derived was “Which are some of the organizations..."
participating in international criminal activity?"

Let’s denote the two sets of questions as REAL and TREC. In our experiment, the REAL queries posed by users could usually be classified as concrete queries— that is, based on specialized knowledge of a domain, while the TREC topics led to more abstract queries.11

Table 4 presents five randomly selected questions from the TREC set and five questions from the REAL set, together with the results obtained. Each table cell contains two numbers: on the top, the number of documents or—for the PARAGRAPH operator— paragraphs retrieved for the question; on the bottom, the number of relevant documents or paragraphs found in the top 10 ranking.

The AND $x_i$ and NEAR $x_i$ columns contain the results for the search when AND and NEAR operators were applied to the input words $x_i$. By replacing the words $x_i$ with their similarity lists derived from WordNet, the number of documents retrieved increased, as expected. The results obtained in these cases are presented in the AND $w_i$ and NEAR $w_i$ columns.

The next column contains the number of documents retrieved when the operator PARAGRAPH 2 (meaning two consecutive paragraphs) was applied to words from the similarity lists. The results were encouraging; the number of documents retrieved was small, and correct answers were found in almost all cases.

Table 5 (next page) presents a summary of results for the 100 questions used to test our system. First, it shows the number of documents retrieved for an average TREC and REAL ques-

---

**Table 4. A sample of the results obtained for randomly selected questions from the TREC and the REAL sets.**

<table>
<thead>
<tr>
<th>TREC questions</th>
<th>AND $x_i$</th>
<th>NEAR $x_i$</th>
<th>AND $w_i$</th>
<th>NEAR $w_i$</th>
<th>Paragraph $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which are some of the organizations participating in international criminal activity?</td>
<td>27,716</td>
<td>3</td>
<td>48,133</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Is the disease of Poliomyelitis (polio) under control in the world?</td>
<td>9,432</td>
<td>13</td>
<td>10,271</td>
<td>15</td>
<td>40</td>
</tr>
<tr>
<td>Which are some of the positive accomplishments of the Hubble telescope since it was launched?</td>
<td>178</td>
<td>4</td>
<td>504</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Which are some of the endangered mammals?</td>
<td>32,133</td>
<td>6,214</td>
<td>32,133</td>
<td>6,214</td>
<td>150</td>
</tr>
<tr>
<td>Which are the most crashworthy, and least crashworthy, passenger vehicles?</td>
<td>246</td>
<td>5</td>
<td>260</td>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REAL questions</th>
<th>AND $x_i$</th>
<th>NEAR $x_i$</th>
<th>AND $w_i$</th>
<th>NEAR $w_i$</th>
<th>Paragraph $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where can I find cheap airline fares?</td>
<td>1,360</td>
<td>3</td>
<td>2,608</td>
<td>35</td>
<td>61</td>
</tr>
<tr>
<td>Find out about Fifths disease.</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>What is the price of ICI?</td>
<td>4,503</td>
<td>202</td>
<td>10,221</td>
<td>575</td>
<td>117</td>
</tr>
<tr>
<td>Where can I shop online for Canada?</td>
<td>36,049</td>
<td>858</td>
<td>36,049</td>
<td>858</td>
<td>15</td>
</tr>
<tr>
<td>What are the average wages for event planners?</td>
<td>6</td>
<td>0</td>
<td>70</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>

---

**<num> Number: 301**

**<title> International Organized Crime**

**<desc> Description:**

Identify organizations that participate in international criminal activity, and, if possible, collaborating organizations and the countries involved.

**<narr> Narrative:**

A relevant document must as a minimum identify the organization and the type of illegal activity (e.g., Colombian cartel exporting cocaine). Vague references to international drug trade without identification of the organization(s) involved would not be relevant.
tion. Naturally, the query extension determined an increase in the number of documents by a factor varying from 1 (meaning an equal number of documents retrieved for both the unextended and extended queries) to 32. Instead of hundreds, thousands, and even tens of thousands of documents, the PARAGRAPH operator returns just 26 and 48 documents for the TREC and REAL questions, respectively.

Moreover, instead of returning full documents, the new operator identifies only the portion of the document where the answer is; this constitutes another reduction factor not captured in the table.

Next, Table 5 shows the precision, or ratio between the number of relevant documents retrieved and the total number of documents retrieved. Because it is impractical to search for the relevant documents among all those retrieved by an AltaVista query, we considered only the relevant documents in the first ten ranked documents. In the case of PARAGRAPH, however, the number of paragraphs retrieved is small, so the precision was considered over the entire set.

With the PARAGRAPH operator, the actual precision reaches 43 percent for the TREC questions and 27.7 percent for the REAL questions. The difference can be explained by the short questions that Internet users tend to ask, which tend to retrieve a very large number of documents and make it much harder to find relevant information.

The biggest gain in Table 5, however, is in system productivity. From the TREC set, 90 percent of the questions were answered correctly; from the REAL set, and 66 percent. This is a significant improvement over current technology.

### Table 5. Summary of results for 50 questions from the TREC collection and 50 questions from the frequently asked queries on the Internet.

<table>
<thead>
<tr>
<th>Number of documents retrieved</th>
<th>AND $x_i$</th>
<th>NEAR $x_i$</th>
<th>AND $w_j$</th>
<th>NEAR $w_j$</th>
<th>Paragraph $w_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average question from the TREC set</td>
<td>7,746</td>
<td>258</td>
<td>25,803</td>
<td>332</td>
<td>26.04</td>
</tr>
<tr>
<td>Average question from the REAL set</td>
<td>13,510</td>
<td>1,843</td>
<td>28,715</td>
<td>3,003</td>
<td>48.95</td>
</tr>
<tr>
<td>Precision</td>
<td>1.6%</td>
<td>4.8%</td>
<td>4.4%</td>
<td>8.8%</td>
<td>43%</td>
</tr>
<tr>
<td>Average question from the TREC set</td>
<td>6.3%</td>
<td>12.43%</td>
<td>6.09%</td>
<td>13.65%</td>
<td>27.7%</td>
</tr>
<tr>
<td>Average question from the REAL set</td>
<td>36%</td>
<td>44%</td>
<td>20%</td>
<td>36%</td>
<td>90%</td>
</tr>
<tr>
<td>Productivity</td>
<td>30%</td>
<td>42%</td>
<td>28%</td>
<td>48%</td>
<td>66%</td>
</tr>
</tbody>
</table>

### CONCLUSION

In general, because the range of questions is so broad, it is difficult to compare the performance of question-answering systems. Other systems implemented for the REAL type of questions operate in narrow domains. For example, Burke, Hammond, and Kozlovsky describe a system that uses the files of "Frequently Asked Questions" associated with many Usenet groups.

The results obtained during the TREC tests can be compared with work described in Voorhees, though the latter retrieves information on very large text collections of texts, rather than the Internet. Voorhees reported an average precision of 36 percent for full-topic statements. Our result of 43 percent precision in retrieving information for narrow questions over heterogeneous Internet domains is thus encouraging.

Our system can still fail to return relevant answers for some questions, for example, questions with very specialized terms. The test results nevertheless demonstrate a substantial increase in both the precision and the percentage of queries answered correctly, while reducing the amount of text presented to the user in comparison with current Internet search engine technology.

The system can be easily extended to restrict the output to several sentences instead of paragraphs. Also, a more flexible NEAR search could be implemented with a new operator $\text{SEQUENCE} (W_1 \text{d} W_2 \text{d} \ldots, W_n)$, where $d$ is a numeric variable that indicates the distance between the words in the $W$ lists for which the search is done.

Indexing words by their WordNet senses, so-called semantic or conceptual indexing, could also improve Internet searches. This method implies some online parsing and word-sense dis-
ambiguation that may be possible in the not-too-distant future. Semantic indexing has the potential for improving the ranking of search results, as well as allowing information extraction of objects and their relationships (for example, see Pustejovsky et al.14).

Finally, Web searches could use compound nouns or collocations. WordNet includes thousands of word groups—for example, blue-collar worker, stock market, and mortgage interest rate—that point to their respective concept. Indexing each compound noun as one term reduces the storage space for the search engine and might further increase the precision.

REFERENCES


Dan I. Moldovan is a professor in the Department of Computer Science and Engineering at Southern Methodist University. His current research interests are in the field of natural language processing, particularly in linguistic knowledge bases, text inference, and question answering systems. Moldovan received a PhD in electrical engineering and computer science from Columbia University in 1978.

Rada Mihalcea is a PhD student in the Department of Computer Science at Southern Methodist University. Her research interests are in the field of natural language processing, particularly in word sense disambiguation, information extraction, and question answering systems. Mihalcea is a member of the AAAI and the ACL.

Readers may contact the authors at {moldovan, rada}@seas.smu.edu.

Next issue in IEEE Internet Computing—
Agent Technology and the Internet

Guest Editors:
Mike Wooldridge and Keith Decker