Question Answering from Frequently Asked Question Files: Experiences with the FAQ FINDER System.

Article *in* Ai Magazine · January 1997

Source: DBLP

6 authors, including:

Robin Burke
DePaul University
194 PUBLICATIONS **9,256** CITATIONS
SEE PROFILE

Kristian Hammond
Northwestern University
173 PUBLICATIONS **4,237** CITATIONS
SEE PROFILE

Vladimir A. Kulyukin
Utah State University
96 PUBLICATIONS **1,438** CITATIONS
SEE PROFILE

Steven L. Lytinen
DePaul University
49 PUBLICATIONS **799** CITATIONS
SEE PROFILE

Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System

Robin D. Burke, Kristian J. Hammond, & Vladimir A. Kulyukin
Intelligent Information Laboratory, University of Chicago
1100 E. 58th St., Chicago, IL 60637
{burke, kris, kulyukin}@cs.uchicago.edu
Steven L. Lytinen, Noriko Tomuro, & Scott Schoenberg
School of Computer Science, DePaul University
243 S. Wabash, Chicago, IL 60604
{lytinen, cphdnt, sschoenb}@cs.depaul.edu

The University of Chicago
Computer Science Department
1100 East 58th Street
Chicago, Illinois 60637

Technical Report TR-97-05

June 1997

Abstract

This technical report describes FAQ Finder, a natural language question-answering system that uses files of frequently-asked questions as its knowledge base. Unlike AI question-answering systems that focus on the generation of new answers, FAQ Finder retrieves existing ones found in frequently-asked question files. Unlike information retrieval approaches that rely on a purely lexical metric of similarity between query and document, FAQ Finder uses a semantic knowledge base (WordNet) to improve its ability to match question and answer.

We describe the design and the current implementation of the system and its support components, including results from an evaluation of the system's performance against a corpus of user questions. An important finding was that a combination of semantic and statistical techniques works better than any single approach. We analyze failures of the system and discuss future research aimed at addressing them.

The University of Chicago Computer Science Department supported this work.

Introduction

In the vast information space of the Internet, individuals and groups have created small pockets of order, organized around their particular interests and hobbies. For the most part, those involved in building these information oases have been happy to make their work freely available to the general public. One of the most outstanding examples of this phenomenon is the wide assortment of frequently-asked question (FAQ) files, many of which are associated with USENET newsgroups.

The idea behind a FAQ file is to record the consensus of opinion among a group on some common question and make that answer available, particularly to newcomers who may otherwise ask the same questions again and again. For this reason, most FAQs are periodically posted on the newsgroups to which they are relevant. This information distribution mechanism works well for individuals who are sufficiently interested in a topic to subscribe to its newsgroup, but not necessarily to those with a more transient interest – having a question about table saws does not necessarily mean one is sufficiently interested in woodworking to read dozens of messages a day about it. What is needed is a centralized means of access to these answers.

We believe that the most natural kind of interface to a database of answers is the question, stated in natural language (Ogden, 1988). While the general problem of understanding questions stated in natural language remains open, we believe that the simpler task of matching questions to corresponding question-answer pairs is feasible and practical. The aim of the FAQ FINDER project is to construct a question-answering system that extends further the aim and intent of the FAQ file phenomenon. The system is an information service, available on the World-Wide Web, to which users can pose their questions. If the question happens to be similar to one of the frequently-asked ones whose answer has been recorded in a FAQ file, FAQ FINDER is likely to return the appropriate answer.

FAQ FINDER is built on four assumptions about FAQ files:

Q&A format: All of the information in a FAQ file is organized in question-answer (Q&A) format. A file is said to be in Q&A format when it is a sequence of Q&A pairs.

Locality of information: All of the information needed to determine the relevance of a Q&A pair can be found within that pair.

Question relevance: The question half of the Q&A pair is the most relevant for determining the match to a user's question.

General knowledge: Broad, shallow knowledge of language is sufficient for question matching.

We see assumptions 1-3 as leading to an essentially case-based (Kolodner, 1993) view of the FAQ retrieval problem. A Q&A pair might be loosely considered a kind of case: it is a piece of knowledge that has been considered useful enough to be codified for reuse. The question serves as an index to the knowledge contained in the answer. These assumptions do not hold for all FAQ files, as we discuss below. However, they hold often enough to form a good starting point for research.

Project history

The FAQ Finder project began in the Spring of 1994. Our starting point was a small set of 50 FAQ files selected essentially at random from RTFM `news.answers`, a large USENET FAQ archive hosted by MIT.[1] These files were manually broken into Q&A pairs for our initial attempts at question matching. At this time, we also gathered sample questions on the topics of the FAQ files from university students to create a corpus of test questions against which the system could be evaluated.

We built several small-scale prototypes and experimented with different matching techniques, culminating in the creation of version 1.0 of the system, which ran more or less continuously as a web resource within the university from May to December of 1996. This version of the system was also demoed at several conferences[2] and described in workshop publications (Burke, Hammond & Cooper, 1996). Local use of the system enabled us to gather a better test corpus. In the Fall of 1996, we undertook to reimplement the system in order to increase its performance and stability for public web release. This version, FAQ Finder 2.0, is now publicly accessible at `<URL:http://faqfinder.cs.uchicago.edu/>`.

Sample interaction

The following sequence of figures depicts a typical interaction with FAQ Finder. Figure 1 shows the World-Wide Web interface to the system. Suppose the user enters the following question:

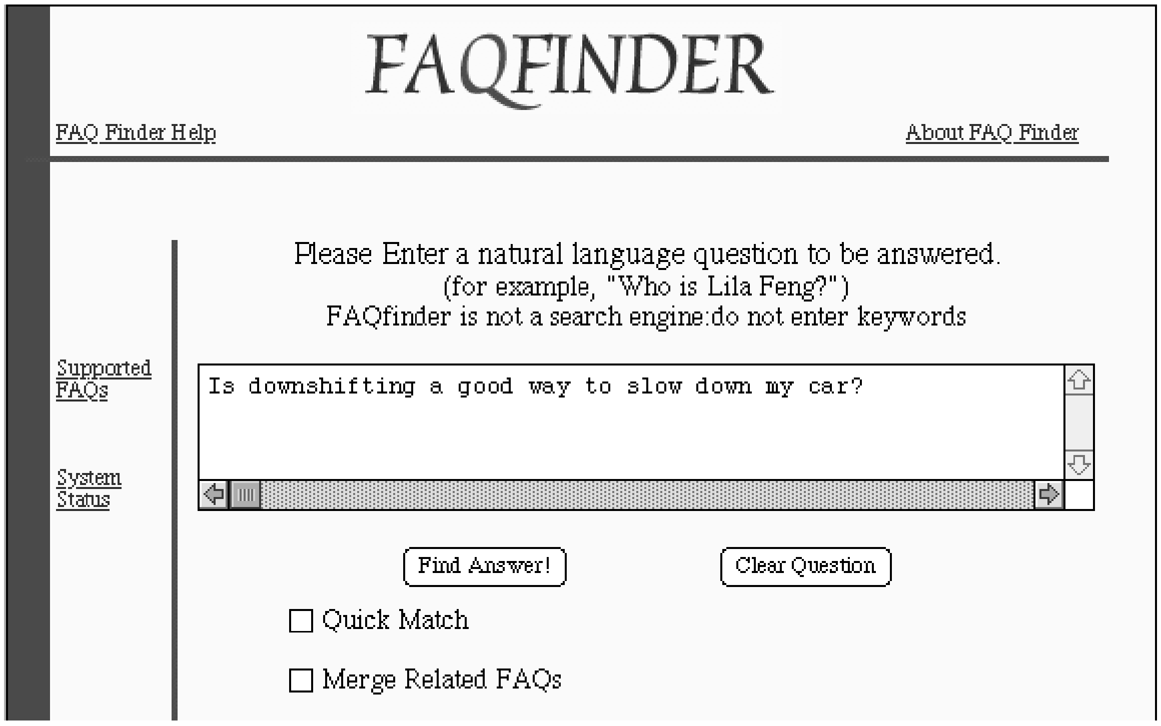
> Is downshifting a good way to slow down my car?

FAQ Finder compares the question to its set of FAQ files, returning a list of files ranked by their relevance to the question. Figure 2 shows FAQ Finder returning the file `auto_consumer_FAQ` as the most relevant file. Some files of questionable relevance are also retrieved, such as the `car_audio_FAQ`, a typical artifact of a keyword-based retrieval system. If the user chooses "Quick Match" when entering a question, the system will skip this first step of manual file selection and automatically choose the top-ranked file to match against.

The user can also choose to "Merge Related FAQ Files" on the question entry page. With this choice, the system will combine related files when retrieving and matching. This is particularly useful when there is a group of closely related files that might be difficult to choose among. For example, for users of Macintosh computers, there are FAQ files for general Macintosh questions, for Macintosh programming, for Macintosh applications, for Macintosh hardware, and for the Macintosh operating system. Suppose three of these files are relevant to the user's query according to the retrieval system. If the "Merge" option is set, the user would only see one entry for Macintosh FAQ files, but the system would match the question against all three.

[1] `<URL:ftp://rtfm.mit.edu/>`

[2] AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval (MIT, 1995), The 5th International World Wide Web Conference (Paris, 1996), AAAI Workshop on Internet-based Information Systems (Portland, 1996)


FAQFINDER

FAQ Finder Help

About FAQ Finder

Supported FAQs

System Status

Please Enter a natural language question to be answered.
(for example, "Who is Lila Feng?")
FAQfinder is not a search engine:do not enter keywords

```
Is downshifting a good way to slow down my car?
```

Find Answer!

Clear Question

☐ Quick Match

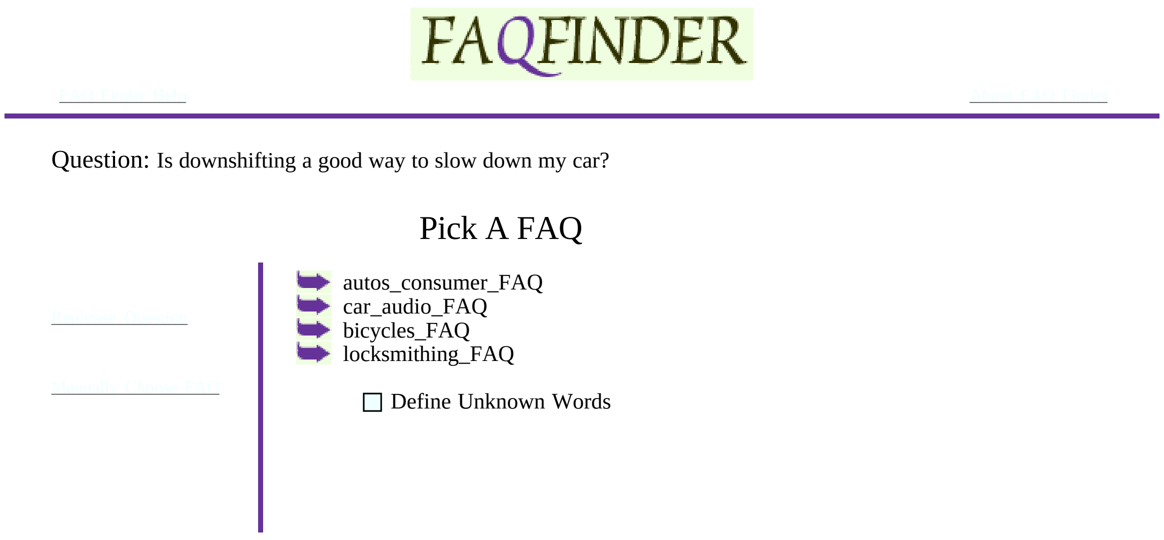
☐ Merge Related FAQs

Figure 1: Asking a question of FAQ Finder.

When a FAQ file is chosen (whether by the system or the user), the system iterates through the Q&A pairs in that file, comparing each against the user's question and computing a score. The best five matches are returned to the user along with the first line of the answer, so the validity of the answer can be easily determined. Figure 3 shows the results of this matching step comparing our "downshifting" question with the entries in the `auto_consumer_FAQ`. The right answer, a question about downshifting and braking is first, followed by two questions about brakes and two about tires.

By selecting the appropriate link, the user can view the entire answer given in the FAQ file as shown in Figure 4.

The remainder of this report has 6 section. In the first section, we give a technical overview of the FAQ Finder system. In the second section, we delve more deeply into the details of the current FAQ Finder implementation. In particular, we dwell on the issues raised by the World-Wide Web as a platform, the necessity to maintain a dynamically changing collection of FAQs, and the need to have a testbed for information retrieval and natural language processing techniques. In the third section, we discuss our evalutation methodology, evaluation metrics, experiments with various aspects of the system and the obtained results. In the fourth section, we analyze the failures of the system and discuss our plans to address them in the future. In the fifth section, we give a description of future work. In the sixth section, we summarize the important points.



Question: Is downshifting a good way to slow down my car?

Pick A FAQ

- autos_consumer_FAQ
- car_audio_FAQ
- bicycles_FAQ
- locksmithing_FAQ

☐ Define Unknown Words

Figure 2: Choosing a FAQ file to match against.

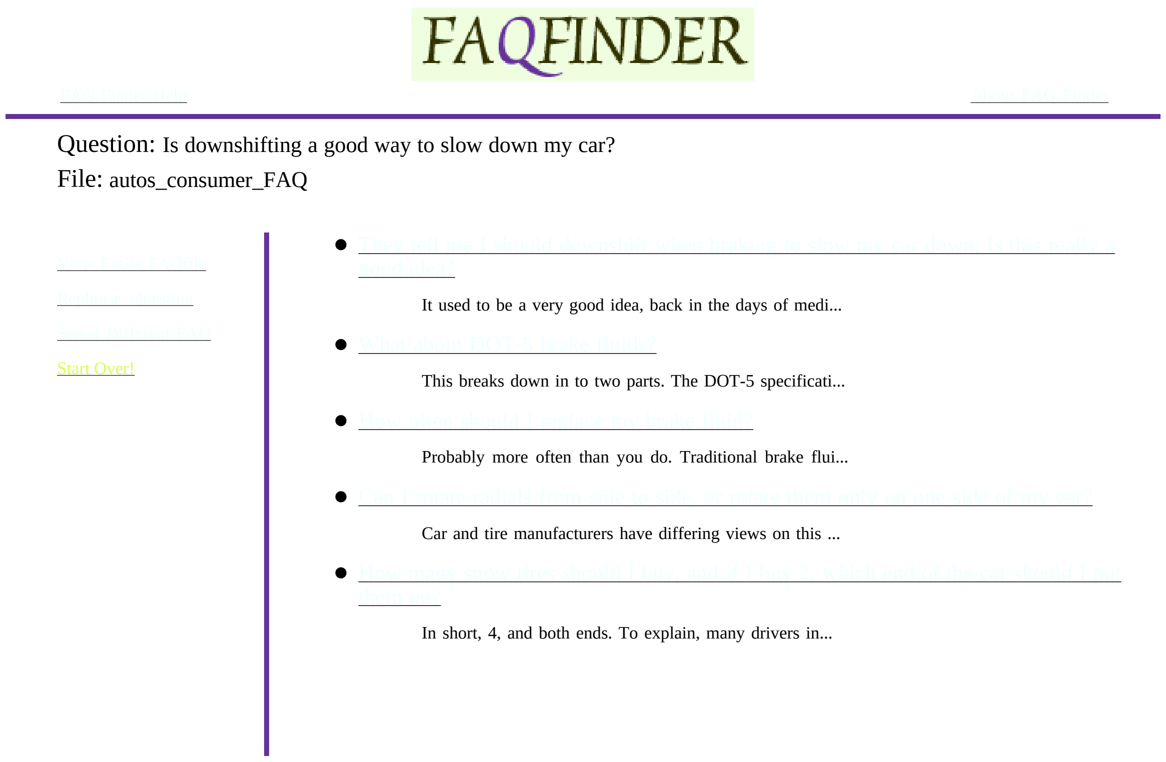
Technical Overview

The diagram in Figure 5 shows that a typical web interaction with FAQ FINDER occurs in a series of stages. The first step is to narrow the search to a small set of FAQs that are likely to contain an answer to the submitted question. The user then chooses a FAQ from the retrieved set to match his question against. The user may choose for the system to do the FAQ selection automatically, in which case the system matches the question against the Q&A pairs in the top ranking FAQ from the retrieved set. Finally, the system's Q&A matcher returns a small set of Q&A pairs to the user.

For the first stage of processing, FAQ FINDER uses standard information retrieval technology, the public domain SMART information retrieval system (Buckley, 1985), to perform the initial step of narrowing the focus to a small subset of the FAQ files. The user's question is treated as a query to be matched against the library of FAQ files. SMART stems all of the words in the query and removes those on its stop list of frequent words. It then forms a term vector from the query, which is matched against similar vectors already created for the FAQ files in an off-line indexing step. We have not found it necessary to tinker with the default configuration of SMART, and treat this part of the system as a black box that returns relevant files.

The second stage of processing in FAQ FINDER is a question matching process. Each question from the FAQ file is matched against the user's question and scored. We use two metrics in combination to arrive at a score for each Q&A pair: a statistical similarity score and a semantic similarity score.[3]

[3]We used to have a third metric, `coverage`, which measured the percent of words in the user's question that the question from the file covers. Our experiments showed that it did not contribute much to the performance of the system, and it is no longer part of FAQ Finder's matching metric.



Question: Is downshifting a good way to slow down my car?

File: autos_consumer_FAQ

- They tell me I should downshift when braking to slow my car down. Is this really a good idea?

  It used to be a very good idea, back in the days of medi...
- What about DOT-5 brake fluids?

  This breaks down in to two parts. The DOT-5 specificati...
- How often should I replace my brake fluid?

  Probably more often than you do. Traditional brake flui...
- Can I rotate radials from side to side, or rotate them only on one side of my car?

  Car and tire manufacturers have differing views on this ...
- How many snow tires should I buy, and if I buy 2, which end of the car should I put them on?

  In short, 4, and both ends. To explain, many drivers in...

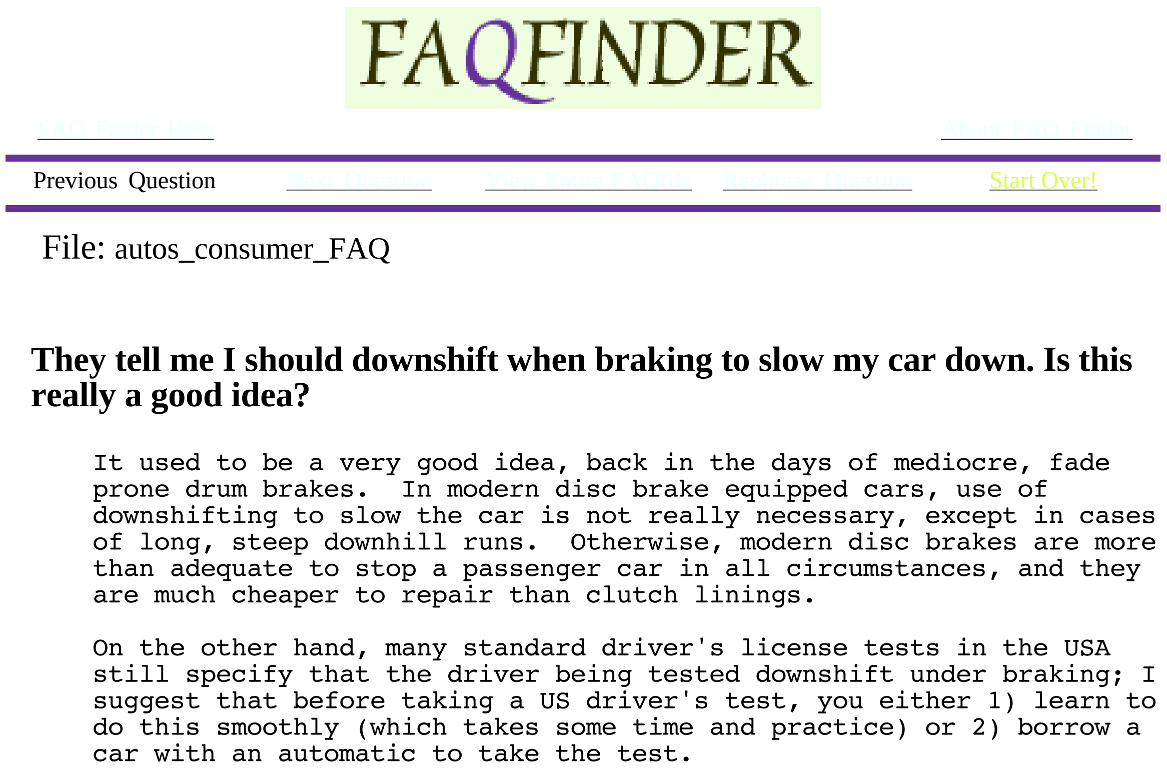
Figure 3: Choosing an answer.

Statistical similarity

The statistical similarity score at the Q&A pair level is computed in a manner quite similar to SMART's document matching. A Q&A pair is represented by a term vector that associates a significance value with each term in the Q&A pair. The significance value that we use is commonly known as *tfidf*, which stands for term frequency times log of inverse document frequency (Salton & McGill, 1983). If n is the number of times that a term appears in a Q&A pair, m is the number of Q&A pairs in the file that contain the term, and M is the total number of Q&A pairs in the file, then *tfidf* is equal to $n \times \log(M/m)$. The idea behind this measure is to evaluate the relative rarity of a term within a space of documents, and use that factor to weight the frequency of that term in a particular document. A term that appears in every Q&A pair in a file is probably of little value, and its *idf*, or $\log(M/m)$ value, would correspondingly be zero. A term that appears in only a single Q&A pair would have the highest possible *idf* value.

Term vectors for user questions are computed similarly by using the *idf* values associated with terms in a given FAQ. Term vectors are then compared using another standard information retrieval metric, the cosine of the angle between the vector representing the user's question and the vector representing the Q&A pair.

The idea behind using the term-vector metric is to allow the system to judge the overall similarity of the user's question and the Q&A pair, taking into account the frequency of occurrence of different terms within the file as a whole. This metric does not require any



File: autos_consumer_FAQ

They tell me I should downshift when braking to slow my car down. Is this really a good idea?

```
It used to be a very good idea, back in the days of mediocre, fade
prone drum brakes.  In modern disc brake equipped cars, use of
downshifting to slow the car is not really necessary, except in cases
of long, steep downhill runs.  Otherwise, modern disc brakes are more
than adequate to stop a passenger car in all circumstances, and they
are much cheaper to repair than clutch linings.

On the other hand, many standard driver's license tests in the USA
still specify that the driver being tested downshift under braking; I
suggest that before taking a US driver's test, you either 1) learn to
do this smoothly (which takes some time and practice) or 2) borrow a
car with an automatic to take the test.
```

Figure 4: An answer returned by FAQ Finder.

understanding of the text, a good thing because the answers in FAQ files are free natural language text, and often quite lengthy.

The *tfidf* measure has a long history in information retrieval and has fairly well-understood properties. In particular, it is generally accepted that the metric works best when queries and documents are lengthy. Only long documents have enough words for statistical comparisons to be considered meaningful. Still, the term-vector comparison works well enough for our purposes in FAQ Finder (see evaluation discussion below), especially when augmented with semantic similarity assessment. Although we have found *tfidf* to be largely reliable, our analysis of the system's failures pointed out some of its shortcomings which we discuss later in the report.

Semantic similarity

We found that statistical matching of questions contributed significantly to FAQ Finder's retrieval, but statistics alone were not enough to achieve the performance we desired. Term-vector comparison suffers from its inability to take into account the meaning of words, relying instead on the global statistical properties of large documents and large queries to ensure that relevant terms will appear. FAQ Finder, on the other hand, deals with small queries and small "documents" – the individual Q&A pairs in each file. The semantic matching algorithm in FAQ Finder is designed to handle variations in lexical content

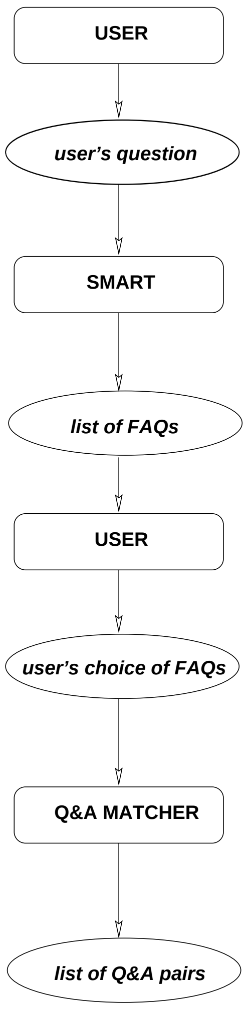


Figure 5: Web Interaction with FAQ Finder.

between input and FAQ questions, variations that might appear naturally in larger corpora. For example, consider the following questions:

How can I get my ex-spouse's debts off my credit report?
Can I get credit if my ex-husband had a lot of debts?

The difficulty in this case is that there are many ways of expressing the same question, all using different words and phrases. The FAQ Finder system needs a means of matching such synonymous but varied inputs against its FAQs. Since the similarity lies in the meaning of these questions, recognizing similarity and matching must make use of *knowledge representation*.

Knowledge representation is a classic AI endeavor. In the FAQ Finder system, it is important to balance the depth of representation with the breadth of coverage. The goal of FAQ Finder is to provide fast answers to an amazingly varied set of questions; deep causal reasoning about questions can be excluded because (1) it would take too long for a quick web interaction, and (2) it would require too much knowledge engineering to cover all of the necessary areas of knowledge.

For FAQ Finder, we believe that a *shallow lexical semantics* provides an ideal level of knowledge representation for the system. Such a semantics has three important advantages:

- It provides critical semantic relations between words;
- It does not require expensive computation to compute relations; and
- It is readily available.

For example, since the consumer credit FAQ file is full of questions about credit reports and debts, it is important that the system identify the relation between "ex-spouse" and "ex-husband." This is the main association between the two question variants. The fact that an ex-husband is an ex-spouse belongs to the lexical semantics of the two words "ex-husband" and "ex-spouse." This is the level of semantics we wish to capture in the FAQ Finder system. We call this a shallow lexical semantics, since it is associated directly with the words.

As an example of deeper semantics, we can consider the following pair of questions:

How do I reboot my system?
What do I do when my computer crashes?

There is a causal relation between the question variants: rebooting is a causal consequent of having one's computer crash. In order to match these questions, the system would have to understand the causality of the computer domain. Since FAQ Finder is intended to encompass the whole gamut of USENET topics, not just computers, it is impractical to expect even this simple level of domain-specific knowledge representation.

FAQ Finder obtains its knowledge of shallow lexical semantics from WordNet, a semantic network of English words (Miller, 1995). The WordNet system provides a system of relations between words and "synonym sets," and between synonym sets themselves. The level of knowledge representation does not go much deeper than the words themselves, but there is an impressive coverage of basic lexical relations.

The WordNet database provides the underlying framework for the FAQ Finder semantic matcher. By using classical marker-passing algorithms, the FAQ Finder system uses the WordNet database to accept variations such as "ex-husband" for "ex-spouse." In particular, we rely on the network's *hypernym* links, which function as *is-a* relationships for nouns and verbs, and the synonym links for adjectives and adverbs. Thus, "ex-husband" and "ex-wife" are semantically related through the common hypernym "ex-spouse."

The matching algorithm we use is based on the classical marker-passing algorithms of Quillian (1968). In Quillian's system, marker-passing in semantic space was used to identify candidate structures which were then compared to *form tests* to judge their linguistic accuracy. For example, the input phrase "lawyer's client" would cause *marker* data structures to be passed through a network from the **lawyer** and **client** concepts. One concept discovered by this search would be the **employment** concept, with the form test: "*first's* `second`." The form test verifies that the input actually was of the proper form to identify the **employment** concept.

From the point of view of FAQ FINDER, Quillian's basic algorithm had the particularly useful feature that it was *fast*. In FAQ FINDER, we are interested mainly in semantic relatedness and not on preserving constraints. The marker-passing phase relies solely on the shallow lexical semantics of WordNet. The relationships between words are not verified.

Score computation

For the purposes of comparing questions, we are interested in arriving at a similarity score using the marker passing technique. Given two questions, we want to know how closely related they are. In other words, we need a function that maps two sets of words to a numerical score. There are many ways that such a function could operate, but we have focused on functions that make all possible word to word comparisons, building a matrix of individual scores, and then reduce that matrix to a single number.

We have designed FAQ FINDER as a modular system so that many scoring functions can be tested. All of scoring techniques we have tested involve building a matrix of scores and then reducing that matrix to a single number. Computing a matrix of scores is one way to capture a word-by-word semantic similarity between every word in a given user's question and every word in a given FAQ question.

The first step in this metric is the word-by-word comparison of questions. Marker passing is performed to compare each word in the user's question with each word in the FAQ file question. Let u_i be the ith word of the user's question. Let f_j be the jth word of the FAQ file question. The similarity score s of these two words is given by

$$s(u_i, f_j) = H - (p\frac{H-L}{D})$$

where p is the length of the path between u_i and f_j and D is the maximum path length permitted by the system. H and L are constants that define the range of s. The score is therefore in the range of H and L, inclusive, and linearly inverse to the number of links traversed between the two words.

For example, "ex-wife" and "ex-husband" have a total of two links between them, up from "ex-wife" to "ex-spouse" and then down to "ex-husband." Under the system's default match scoring scheme, their match score is 0.24. "Ex-husband" and "ex-spouse" which are separated by a single link have a score of 0.32.

No marker passing is necessary when there is an exact string match between a word in a user's question and a question in a FAQ. Nor does one need to pass markers when the two words are morphologically related, i.e. when morphological analysis reduces both of them to the same word. For example, the words "easiest" and "easily", which reduce to "easy", are morphological relatives. We give fixed scores to words that are identical strings and morphological variants, preferring identical matches to morphological variants.

The matrix S for a user question of length n and a FAQ file question of length m is an $n \times m$ matrix:

$$S_{u,f} = \begin{bmatrix} s(u_1, f_1) \cdots s(u_1, f_m) \\ \vdots \\ s(u_n, f_1) \cdots s(u_n, f_m) \end{bmatrix}$$

This matrix S is reduced to a single value, w, by choosing the maximum match score for each user question and then averaging these maxima for all words. The value $w(u, f)$ for semantic relatedness of the two questions u and f is given by

$$w(u, f) = \frac{\sum_{i=1}^{n} \max(s(u_i, f_1), ..., s(u_i, f_m))}{n}$$

Finally, we combine the two scores: the term vector similarity value t and the semantic similarity value w in a weighted average to arrive at an overall match score m.

$$m = \frac{tT + wW}{T + W}$$

where T and W are constant weights associated with term vector and WordNet scores, respectively.

Implementation Issues

Interfacing to the World-Wide Web

Several aspects of FAQ FINDER's design are a direct consequence of its targeted environment: the World-Wide Web (WWW). As is now well understood, HTTP, the WWW protocol, is essentially stateless. Any system state that must be maintained by FAQ FINDER comes at a cost of transmission overhead.[4] For this reason, our system is designed much like a web-based "search engine." It does not engage in an extended interaction with the user over time, but works on the model of "question in, answer out." Often we must ask a clarifying question – the specification of which file to match against – but other than that the interaction is very simple.

In some ways, the use of the Web simplifies the system's design: many user interface decisions are already made, but it also reduces the designer's flexibility. AI programmers have traditionally profited from the rapid prototyping environment provided by Common Lisp, and we sought to maintain this flexibility for our work with FAQ FINDER. Our initial implementations used the Common Gateway Interface (CGI) protocol from a standard web server. However, we found that a substantial portion of the system's response time was consumed by system processing tasks, in particular, the spawning of Unix processes and the creation of TCP connections. In order to interface transitory web requests with a running

[4]Other state management techniques, such as "cookies" are available, but not supported by all browsers and may be disabled by the user. We seek to make FAQ FINDER useful for as wide an audience as possible, and so are avoiding the cookie mechanism where possible.

Common Lisp image, we needed intermediary programs handling socket traffic, which made the system essentially single-user: no processing could be done on a question that arrived at time $t + 1$ until the question submitted at time t was completely processed. This mode was highly inefficient since, as we discuss below, much of FAQ FINDER's processing time is spent in file input.

Our solution to these difficulties was to use the Common Lisp-based web server CL-HTTP[5] running under Allegro Common Lisp. Multi-processing is handled within Common Lisp itself, eliminating the need for operating-system level startup and shutdown of processes. Efficient multi-user processing is a natural consequence. In addition, the CL-HTTP server provides a mechanism natural to the Common Lisp programmer for the implementation of dynamic web content and the handling of HTTP requests: A URL can be associated with a function, and the access of that URL becomes a function call.

File management

FAQ FINDER matching algorithm compares user questions against Q&A pairs from the FAQ file. To identify the Q&A pairs that comprise a given file, the system needs to have a structural understanding of FAQ files, equivalent to the quick visual scanning of a screen or page that enables a human reader to distinguish between different structural elements. Unfortunately, writers of FAQ files have human visual capacities in mind when structuring their files, and are not necessarily consistent or regular in their structuring devices. Therefore, the task of "tagging," identifying important structural elements in FAQ files is an important research area within the FAQ FINDER project.

We have experimented with various techniques for question and answer tagging. The FAQ MINDER system was developed as a comprehensive approach to analyzing text file structure (Kulyukin, Hammond & Burke, 1996), and in its semi-automated mode is our most accurate tagging technique. However, we reserve this labor-intensive process for only those files with obscure or irregular organization. The majority of FAQ FINDER files are tagged automatically using a regular expression-based perl program called FAQ GRINDER.

On regular FAQ files, FAQ GRINDER system achieves about 90% of the number of Q&A pairs as hand-tagging. However, this does not necessarily reflect the accuracy of the automatic tagging: spurious question patterns caused both false positive and false negative taggings.

False positives occurred when a non-question pattern in one file matched with a question pattern in another file, and this pattern occurred more frequently than the real question pattern in the file. This was caused mostly because of less specific patterns. For example, in the `hp_hpux-faq`, the following line was tagged as a question:

```
:QUE
Subject: 3.2  Courses on HP-UX
:QUE
```

[5]<URL:http://www.ai.mit.edu/projects/iiip/doc/cl-http/home-page.html>

This line should be a topic instead of question in file, since the real questions are listed one level below the index number: for example, "Subject: 3.2.1. What courses are available for HP-UX?"

The false negative case happened when the patterns discovered so far are more specific than the real question pattern in a file. For example, in `woodworking-faq`, the following question line was missed:

```
16). Which saw blade should I buy?
There is an excellent article on evaluating carbide tipped
sawblades in issue #72 of Fine Homebuilding (March 1992).
```

All of the other questions in the file have a blank line between question and answer, but a typo here prevented recognition.

This FAQ GRINDER system classifies the FAQ files according to the question patterns through iterative matching process. As a pattern is identified for one untagged file, it is matched against all other untagged files, thus all the files which fall under that pattern category are discovered. By repeating this process, FAQ files are incrementally classified by their question patterns.

The FAQ GRINDER system is both effective and efficient: the question part of Q&A pairs in FAQ files can often be represented by regular expressions, and the iterative classification takes advantage of the commonality among question patterns in FAQ files.

FAQ files are frequently updated. In its production mode, FAQ FINDER updates its set of FAQ files weekly. It is essential therefore to automate the process of integrating new files in the system as much as possible so that the system can remain up-to-date indefinitely. We are using the "mirror" package[6] for keeping our files in sync with those at the RTFM archive. The files at the RTFM `news.answers` site are arranged in a system of directories analogous to the USENET hierarchy, with long files divided into 64K segments. For our purposes, we need a flat directory of files and we want long files whole, not in parts. A flattening and concatenating step must therefore be performed on the mirror directory structure. As of this writing, this step yields 2041 distinct files that are potential FAQ files.

Many of the files in the `news.answers` are not in Q&A format. These non-FAQ files fall into several categories:

- Long text discussions of a topic. For example, a description of the characteristics of a particular breed of dog: `dogs-faq_breeds_bassets`.
- Lists of addresses, such as a list of bookstores that handle mail orders: `books_stores_ship-by-mail`.
- Tables of facts. For example, a table listing every geographical place name that has ever appeared in a Star Trek episode: `Star-Trek_locations`.
- Regularly posted "diff" files that indicate what has changed in a long FAQ file, e.g. the file `GNU-Emacs-FAQ_diffs`.

[6] `<URL:ftp://src.doc.ic.ac.uk/computing/archiving/mirror/>`

These files do answer questions that are frequently-asked. The `dogs-faq_breeds_bassets` file is a perfect answer to a question like "Are bassets good with children?" However, we have opted to concentrate on files where the questions being answered are made explicit by the author, because it is in these files that we can use the question as an index.

The determination of whether a file is in Q&A format is something that we are not yet able to automate. We have manually labeled all 2014 files in the `news.answers` archive, finding that about 50% of the files are in Q&A format. We expect that automated updating can make use of our existing determinations about the format of existing files. Newly-created files will have to be examined individually to decide whether to include them or not. Our automatic tagger can give a good indication of whether a file is likely to be in Q&A format or not, but in many cases, manual examination will still be required.

Preprocessing FAQ files

FAQ FINDER's matching process uses data specific to a given FAQ file for much of its work. Obviously, it is matching against the specific Q&A pairs in the file, but there is also a file of phrases and a table of *idf* values specific to the FAQ file. FAQ FINDER's matching process must also read in the Q&A pairs of the FAQ file to perform its comparison. As a result, the matching process is I/O intensive. It is this aspect of the system that we have tried to optimize as much as possible.

Our general strategy has been to perform as much processing as possible off-line, leaving only those parts of the process that depend on the user's input to be performed at run time. One of the largest gains in efficiency was achieved by the pre-processing of the FAQ files themselves. Internally, FAQ FINDER has a representation of the user's question and the FAQ file Q&A pair, called a "question text" or "qtext" representation. The qtext representation for a Q&A pair consists of the following parts:

Text: The raw text question from the file.

Words: A list of word data structures representing the question. These data structures each contain a word, its morphological reduction, a part-of-speech tag, and a pointer to the word's place in the WordNet semantic network.

Vector: A term vector representation of the Q&A pair in *tfidf* form.

Answer-tag: The first 60 characters of the answer to the question for output to the user as in Figure 3.

Location: An index to the location of this Q&A pair within the FAQ file.

Note that the answer is represented only within the "vector" part of the qtext representation. Since answers are generally longer than questions and can run into the thousands of words, a qtext representation of a Q&A pair is considerably smaller than the text it represents. In addition, the creation of the "words" slot in a qtext representation requires considerable computation: dividing the string into words, computing morphological variants, looking up part-of-speech and WordNet information. The qtext representation of a

FAQ file therefore represents not only a compressed version of the file, but one in which the results of considerable computation are stored. We build a qtext representation of each Q&A pair in each FAQ file and store them in an associated file, for matching against user questions. The user's question must be converted into the qtext representation for matching to take place, but this only happens once per query and the only cost incurred per Q&A pair is the reading and instantiation of the qtext data structure.

Precompiling WordNet

Another significant efficiency gain was achieved by rebuilding the WordNet semantic network. WordNet, especially in its new 119,000 word incarnation, is too large to keep in core memory all at once. However, much of WordNet is unnecessary for our purposes. The system includes associations between antonyms, part/whole relations, and other relations that we do not use. All FAQ Finder needs is what words are linked to others via the hypernym and synonym links. We used WordNet to build a "tree" dictionary: associated with each word is a tree of hypernyms and synonyms. For example, the entry for "wife" in this dictionary is

```
(wife ((woman (female (person ((life_form (entity ()))
                               (causal_agent (entity ())))))))
      (spouse (relative (person ((life_form (entity ()))
                                 (causal_agent (entity ()))))))))
```

With these trees, marker passing is reduced to identifying the topmost level at which two such trees have a common element. The dictionary of these trees is actually much larger than the WordNet dictionary, since information is recorded redundantly. However, only a single read is required to access the entire tree, so retrieval is at least as efficient as using the WordNet executables in the running Lisp image. As part of the pre-processing of FAQ files, we also record the offset into the tree dictionary file for each word in each FAQ file question. At run time, the only lookup that is required is for the words in the user's question.[7]

Another important use of WordNet is in the compilation of FAQ-specific phrase vocabulary. WordNet contains many phrases in its sets of synonyms, making it possible to identify connections not just between single words but between groups of words like, for example, "table saw" and "power tool." Such phrases provide more specificity. In order to make use of phrases, the part of FAQ Finder that breaks text into words needs to consider the possibility that a set of words should be treated as a single phrase. Again, the phrase dictionary in WordNet is too large to keep in memory all at once, so in an off-line step, we build a subset of the phrase dictionary that is specific to a given FAQ file, identifying all phrases that are used in that file. The phrase file is loaded with the other FAQ-specific auxiliary files during the matching step. FAQ-specific phrases are therefore present in the system's memory when the user's question is examined and can be recognized.

[7]We could further simplify our representation by eliminating upper levels of the tree never reached by the depth-constrained marker passing but for the moment we keep the maximum amount of information so that we can experiment with different uses of WordNet.

FAQ Finder's Configurations

Since the very beginning of the FAQ FINDER project our priority was to implement an easily extensible system that can serve not only as a useful web resource but also as a testbed for information retrieval and natural language processing experiments. The focus on this priority lead to the notion of configuration that completely determines the behavior of the system and yet is flexible enough to accomodate changes whenever necessary. To achieve this objective, we have implemented a CLOS class named `configuration`. Along with the hash tables and path names, the class contains all of the matching functions used during the indexing of FAQ files and the matching of questions. A researcher can switch from one scoring function to another, adjust an existing function's parameters, or define and plug in a new function. All modules of FAQ FINDER communicate with each other through the configuration defined at startup.

Hierarchy of Requests

A user's interaction with FAQ FINDER can be construed as a sequence of requests to the system. After the question is typed in, the system is requested to identify a small set of relevant FAQs. When the user chooses a FAQ, the system is requested to find and rank the Q&A pairs in that FAQ that are closest to the question. The preprocessing stage during which the raw text FAQs are turned into the qtext objects is also a sequence of requests to the system: tag a FAQ, index it with SMART, find WordNet phrases, do the morphological analysis of each Q&A pair, compute the *tfidf* table for the FAQ treating each Q&A pair as a separate document, combine the results of the previous steps into the qtext representation, and, finally, write the qtext objects into the database of persistent CLOS objects.

Our objective was to ensure that researchers can experiment with different behaviors of the system. The most natural way to do so was to give them the option to modify what happens in any of the above steps. Since FAQ FINDER consists of several modules that constantly communicate with each other, it was imperative that intermodular communication be abstract so that a modification in one module would not percolate into the dependent modules. On the conceptual side, we wanted FAQ FINDER to accurately model the world of requests in which it functions.

Given these considerations, CLOS, the Common Lisp Object System, was a natural choice. We used CLOS to implement a hierarchy of request classes. The `REQUEST` class on top of the hierarchy consists of the following parts:

Handlers: A list of functions to handle low level computations. One of the generic functions, `REQUEST-HANDLE`, goes through the list and calls each of these functions. Different request classes have different sets of handlers.

Configuration: The configuration of the system. Each request must be aware of how the system has been configured to behave. For example, the `MATCH-REQUEST` class, which computes the QA matches to the user's question, depends heavily on the question matching function specified in the configuration.

All of the other request classes specialize the `REQUEST` class. The full hierarchy is given in Figure 6. The hierarchy combines three types of requests: preprocessing requests, matching requests, and test requests. Since these types are very broad and generic, we have not implemented them as separate classes. We have found it handy, however, to use them in describing how requests behave.

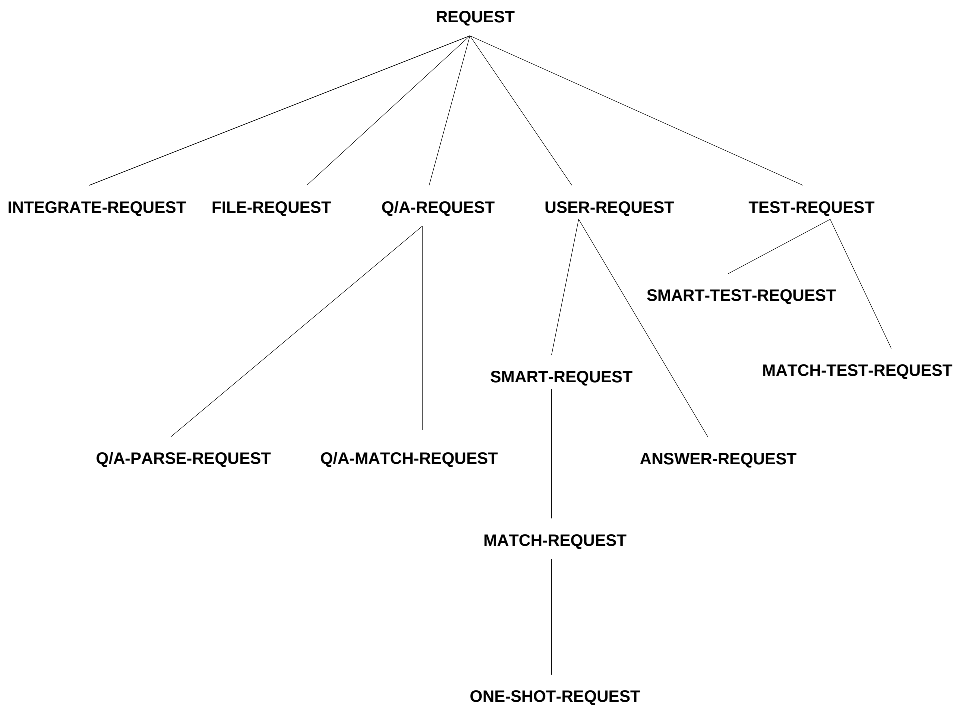


Figure 6: A Hierarchy of Requests in FAQ Finder.

Preprocessing Requests

The preprocessing requests are used by the system during the off-line stage when FAQs are turned into qtext objects. The `INTEGRATE-REQUEST` is issued when a raw FAQ needs to be integrated into the existing collection of processed FAQs. It tags the FAQ, identifies WordNet phrases, turns the FAQ into a set of qtext objects, and reindexes the new collection with SMART. The `INTEGRATE-REQUEST` interacts with the `FILE-REQUEST` and the `Q/A-PARSE-REQUEST`. In brief, the `FILE-REQUEST` gathers all of the pieces of data that FAQ Finder needs to know about a particular FAQ. The `Q/A-PARSE-REQUEST` turns a Q&A pair into a qtext object which it stores in the database of persistent CLOS objects.

Matching Requests

The matching requests are issued during the question answering stage. Generally speaking, the `USER-REQUEST` handles the situation when the user asks FAQ Finder to do something.

After the user types in a question, the `SMART-REQUEST` is issued to FAQ FINDER to return a small set of FAQs that are statistically most relevant to the question. After the set of FAQs is retrieved and the user selects one of them, FAQ FINDER handles the `MATCH-REQUEST`. The `Q/A-MATCH-REQUEST` computes the similarity score between a qtext object that represents a Q&A pair and the user's question. The execution of the `MATCH-REQUEST` returns a list of 5 questions ranked by the similarity score. When the user clicks on one of them, the `ANSWER-REQUEST` displays the answer to the question given in the FAQ.

The user can choose to bypass the manual selection of a FAQ by selecting the quick-match option on the interface. When the option is selected, the `ONE-SHOT-REQUEST` is issued. It automatically assumes that the top FAQ returned by the `SMART-REQUEST` is the most relevant, and runs the `MATCH-REQUEST` on it.

As FAQ FINDER progresses, it will need more types of user interactions. We expect the `USER-REQUEST` class will have many more specializations in the future.

Test Requests

The purpose of test requests is to allow researchers to define and run experiments on various components of the system. The `SMART-TEST-REQUEST` was defined to run experiments on how accurate the SMART system is at retrieving the FAQs relevant to user questions. The `MATCH-TEST-REQUEST` was defined to test various matching functions and configurations of the system. The difference between the regular requests and the test requests is that the latter collect statistics about the system's performance in addition to doing the work of the regular requests.

Dynamic Database of FAQs

The collection of FAQs used by FAQ FINDER is dynamic: a new FAQ can be added at any moment and an old FAQ can be modified by its maintainers. To handle such changes we are turning our original ideas about FAQ Minder and FAQ Grinder into a comprehensive Web-based tagging and database utility: FAQ Minder 2.0. It has at its disposal a library of regular expressions, or patterns, that identify Q&A pairs in FAQs. When a new FAQ arrives, FAQ Minder checks it against each pattern looking for the best fit. Before the file is fully integrated into the system, the FAQ Finder system's administrators have the option of looking at the tagged and untagged versions of the FAQ and giving recommendations to it. If there is a fit between a pattern and a FAQ, the FAQ is added to the database and processed in the manner described above. When FAQ Minder fails to find a fit between the new FAQ and any of its patterns, a new pattern may need to be entered into the system through a simple interface.

Once a FAQ has been successfully tagged, the pattern that tagged it is stored in the database. So when the FAQ is updated, it is easy for FAQ Minder to reintegrate it into the system. This approach is based on the assumption that once the author of the FAQ has chosen a specific way to mark questions and answers, he or she is likely to stick with it.

Experiments

Before the creation of the current version of FAQ FINDER, the system went through many different incarnations as we tested various matching and indexing techniques. This section describes our evaluation methodology and metrics as well as the aspects of the system we experimented with.

Evaluation methodology

We evaluated the performance of FAQ FINDER on a corpus of 241 questions drawn from the log files of the system's use during the period May to December of 1996.[8] We manually scanned each FAQ file for answers to each question, and determined that there were 138 questions that had answers in the FAQ file corpus, and 103 questions that were unanswered. There are hundreds more questions in the system logs awaiting incorporation into our testing regime.

Evaluation metrics

The most obvious precedents to FAQ FINDER are information retrieval systems, and standard information retrieval evaluation techniques are a starting point for the evaluation of the system. However, evaluation in FAQ FINDER is complicated by the fact that the task of the system is different than the information retrieval problem as it is typically posed. Normally, the assumption is that there is a document collection in which there may be a number of documents relevant to the users' query. It is the system's job to return as many of these relevant documents as possible. In contrast, FAQ FINDER works under the assumption that there is such a thing as a "right answer": an answer that best addresses the user's question as it was posed. The system's job is to return that answer within the small fixed-size set of results that can be displayed on a single web page. Relevance is not that useful a measure to us because, within a given FAQ, many answers are likely to be somewhat relevant to the user's query.

Because of the differences in the FAQ FINDER task, the traditional IR evaluation metrics of recall and precision must be modified somewhat. Recall normally is a measure of the percentage of relevant documents in the document set that are retrieved in response to a query, whereas precision is a measure of the percentage of retrieved documents that are relevant. In our case, however, since there is typically one right answer to be retrieved from a FAQ, these are not independent measures of performance. Assuming that an answer to a user question exists in a FAQ file, FAQ FINDER will perform at either 100% recall and 20% precision (if the answer is retrieved), or 0% recall and precision (if it is not). If no answer exists, then precision will be 0%, and recall is undefined.

To measure the quality of retrieval, we calculate our version of recall, which amounts to the percent of questions for which FAQ FINDER returns a correct answer when one exists.

[8]Our earlier evaluation showed a significant difference between the quality of retrieval on our initial question corpus, gathered by email survey, and on the questions gathered from the logs of web interactions. The email questions were more syntactically complex, and more difficult to answer than the questions typically submitted from the web page. Performance on email questions was about 7% worse overall.

Our calculation is different from traditional recall measures, because it does not penalize the system if there is more than one right answer in the file. If there are several answers within a file that answer a user's question, it does not make sense to regard retrieval of only one of these answers as partial success. If the user's question is answered, it is irrelevant that there was another Q&A pair that also answered it. Instead of precision, we calculate a value called *rejection*, the percentage of questions that FAQ FINDER correctly reports as being unanswered in the file. We feel that these metrics better reflect FAQ FINDER's real-world performance than traditional recall and precision would.

Rejection is adjusted in FAQ FINDER by setting a cut-off point for the minimum allowable match score. As with precision, there is a trade-off between recall and rejection rate. If the rejection threshold is set too high, some correct answers will be eliminated; on the other hand, if the threshold is too low, then garbage responses will often be given to the user when no answer exists in the FAQ file.

Morphological Analysis

Unlike most information retrieval systems, FAQ FINDER does morphological analysis. We decided to do morphological analysis instead of stemming for two reasons. First, stemming algorithms often conflate semantically different words into one single stem. For example, "informative", "informant", "informed", "information" are reduced to "inform." Since FAQ FINDER attempts to capture not only statistical but also semantic similarities, such conflation is not desirable. Second, the semantic hierarchies of WordNet contain actual words, not stems. Most words in WordNet are given in their basic forms, i.e. nouns in the singular, verbs in the infinitive, etc. Thus, in order to use the semantic relations of WordNet, FAQ FINDER has to convert the words in user questions and in FAQs to their basic forms.

To do morphological analysis, we use a rule-based algorithm which searches a list of ending rules. An ending rule has the form `(match-pattern suffix-id add-pattern)`, where `match-pattern` and `add-pattern` are strings and `suffix-id` is a symbol like `ING` or `ED`. The match-pattern determines whether the rule applies to a word. If the rule is applicable, the match-pattern is stripped off the end of the word and the add-pattern is added on. For example, the rule `(''ily'' LY ''y'')` matches the word "easily". As a result, "ily" is stripped off and "y" is added onto "eas" giving "easy". Sometimes we have to add back characters from the match pattern. For instance, the words "racing" and "glaring" are handled by the rule `(''@?ing'' ING ''01e'')` , where the sign `@` matches vowels and the sign `?` matches any character. After the match-pattern is stripped off, the word "racing" turns into "r" while the word "glaring" into "gl". Then the first and the second characters of the match-pattern and the vowel "e" are added back on. Our current database consists of 32 such rules for nouns, verbs, and adjectives.

Anyone who attempted a comprehensive formal analysis of English morphology knows how hard it is to write elegant rules that handle regularities as well as exceptions. The rules quickly become involved and expensive to process. Since morphological analysis is performed on every user question, it has to be fast. While a comprehensive set of morphological rules for English remains an open research problem for computational linguistics, we have found that many irregular verbs can be satisfactorily handled by a simple table lookup. Toward that end, we have compiled a list of 226 most common irregular verbs on the basis of A. S. Hornby's Oxford Student's Dictionary of Current English (Hornby, 1984). Thus, before the 32 rules are applied, the word is looked up in the table of irregular verbs. We plan to compile a similar list of irregular adjectives like "good", "better", "best", etc.

Restricting marker passing

WordNet is not a single semantic network; separate networks exist for nouns, verbs, adjectives, and adverbs. Syntactically ambiguous lexical items, such as "name," which could be either a noun or a verb, appear in more than one network. We found that unrestricted marker passing, using all networks in which a term appears, led to too many spurious matches, a common problem in marker passing systems in general (Collins & Quillian, 1972).

We tried several approaches to disambiguate terms to a single WordNet network. Our first attempt was to use an existing part-of-speech tagger, the Xerox Tagger (Cutting, et al., 1992). This system uses a hidden Markov model learned from a large corpus of English text to statistically determine the most likely sense for any given word in the context of a sentence. The system worked well in many cases, but it had a significant drawback in that it was not robust in the face of unknown words. By default, it assumes that unknown words are nouns. Since unknown words are unlikely to appear in WordNet anyway, this is not an immediate problem. However, the tagger uses its determination that the unknown word is a noun to influence the rest of its tagging. For example, the word "reboot" is unknown to the tagger. When tagging the sentence "How do I reboot faster?" the system does not mark "faster" as an adverb because there is no verb for it to refer to. "Faster" is marked as a noun, presumably meaning a person who fasts, which is an inappropriate word sense in this context. Because FAQ files contain technical vocabulary from many different fields, and therefore many terms that are unknown to the tagger, we found we could not use this method for disambiguation.

Our next attempt at identifying unambiguous part-of-speech information was to use natural language parsing. We built a simple context-free grammar for questions, and implemented it in a bottom-up chart parser. The parser's lexicon was compiled from a combination of the on-line Oxford English dictionary and the Moby part-of-speech dictionary (Grady Ward, 1994). A successful parse would resolve syntactic category ambiguities of individual lexical items in a question (e.g., whether "name" in the above example is a noun or a verb). If the parser finds more than one parse for a question, the parses are ranked according to word sense frequency information from the on-line dictionaries, and the top parse selected.

Analysis of the effects of parsing on FAQ Finder performance showed that the desired goal of improving semantic matching was not achieved. We compared a version of the system using parsing with one in which lexical items which appear in the on-line dictionaries were tagged according to their statistically most frequent word sense. Our analysis indicated that parsing did not have a significant effect. One reason for that is that most questions in our sample set did not have severe cases of morphological ambiguity. Therefore, our simple part-of-speech tagging turned out to be highly accurate and reliable. We cannot, however, conclude from this that parsing is useless. Rather, the necessity to parse should be determined by the character of the questions a particular system must handle.

Parsing for question type

Another area where we devoted significant research attention was the issue of question type. Our intuition was that abstract categories of questions could be a useful part of the matching equation. For example, a "temporal" question from the user should probably match against a "temporal" question in a file as opposed to a "how to" or "where" question on the same topic.

We used the same parser that was employed for part-of-speech disambiguation to determine question type. The grammar included question types as nonterminal symbols in the grammar; thus, categorization of a question occurred automatically as a by-product of parsing. The grammar included nonterminals for `Q-WHAT`, `Q-HOW`, `Q-ADVICE`, `Q-COST`, `Q-LOC`, `Q-TIME`, `Q-WHY`, `Q-YES-OR-NO` and other subtypes of these question types. The final grammar included rules such as the following:

```
S      -> Q-HOW | Q-WHAT | ...
Q-HOW  -> How do NP VP | ...
Q-WHAT -> What is NP | ...
```

Generic semantic cases were computed from the parse tree using a set of rules which mapped syntactic roles to semantic cases. For instance, in a Q-HOW question, the NT directly following "How do" was assigned the ACTOR/AGENT case, while the direct object of the verb the OBJECT/PATIENT case. Objects of prepositions were assigned very general cases so as to allow for variations in use of prepositions across questions.

What became immediately obvious from our implementation, however, was that syntactic information was insufficient to determine abstract question types and cases with any reliability. The question "How often should I change the oil on a shovelhead Harley?" can be easily identified as having the `Q-TIME` type by the phrase "How often," but the same question can be easily rephrased into a different syntactic type without changing its meaning "What is the proper oil change interval for a shovelhead Harley?" Our parser would categorize this as a different syntactic question type, `Q-WHAT`. It is an open question what other techniques might be employed to more accurately assess semantic question type, possibly using categories such as Lehnert's (1978). We are continuing to hold this option open as an area for future research.

Recall vs. rejection

We concentrated our evaluation efforts on the question-matching portion of the system because we found SMART to be highly effective at the file retrieval task.[9] Our tests were

[9]The correct file appears 88% of the time within the top five files returned to the user, and 48% of the time in the first position. Using standard IR metrics, it would stand at 88% recall and 23% precision.

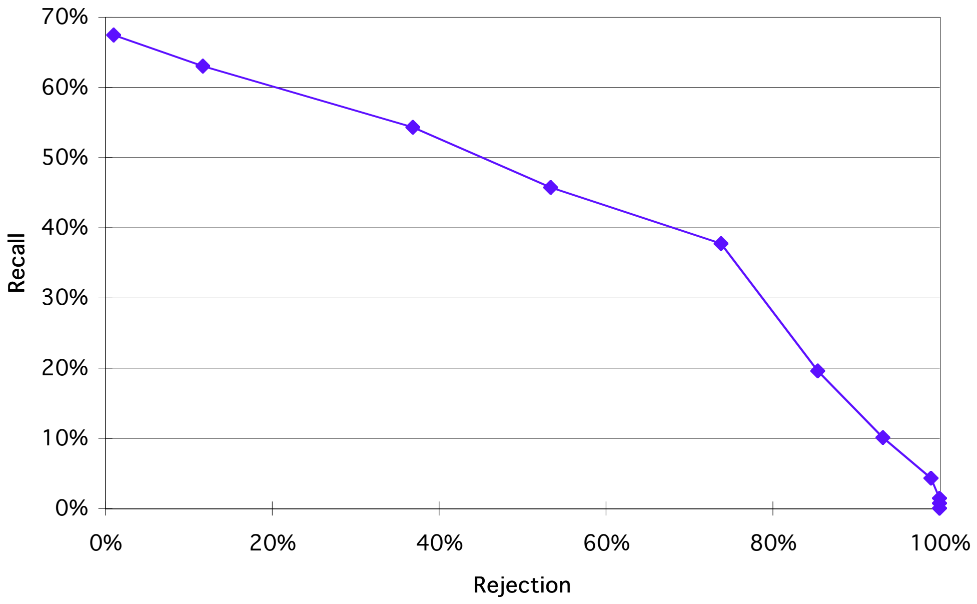


Figure 7: Recall vs. rejection for FAQ Finder

performed assuming that the correct file had been selected. If we were not to make this assumption, the system overall performance would obviously be worse, but since the user gets to select an appropriate file, he or she can usually identify the case in which none of the retrieved files are relevant.

Figure 7 shows the recall vs. rejection results that we obtained for the FAQ Finder system. As the graph shows, rejection is disappointingly low for reasonable values of recall, meaning that the system confidently returns garbage in most cases when there is no right answer in the file. If the rejection threshold is lowered to make it easier to identify questions without good answers, recall drops dramatically. However, the top value for recall is encouraging: better than a two-thirds probability that the system will find the right answer.

Ablation study

Our next step was to evaluate the contribution of different components of the matching scheme through an ablation study. We selectively disabled different parts of the system and ran the same corpus of questions. There were four conditions: a *random* condition, in which Q&A pairs were selected randomly from each FAQ file; a *coverage only* condition, in which the coverage score for each question was used by itself; a *semantic score only* condition, in which only the semantic scores derived from WordNet were used in evaluating answers; and, a *statistical score only* case, in which the term vector comparison was used in isolation.

Figure 8 shows average recall results for these conditions. Interestingly, both WordNet and our statistical technique are contributing strongly to system performance. Coverage score, which is an extremely weak measure in isolation, turned out even worse than selecting questions randomly, which was one of the reasons why we stopped using it. Semantic scoring and statistical scoring had very similar average recall, but are clearly equivalent measures,

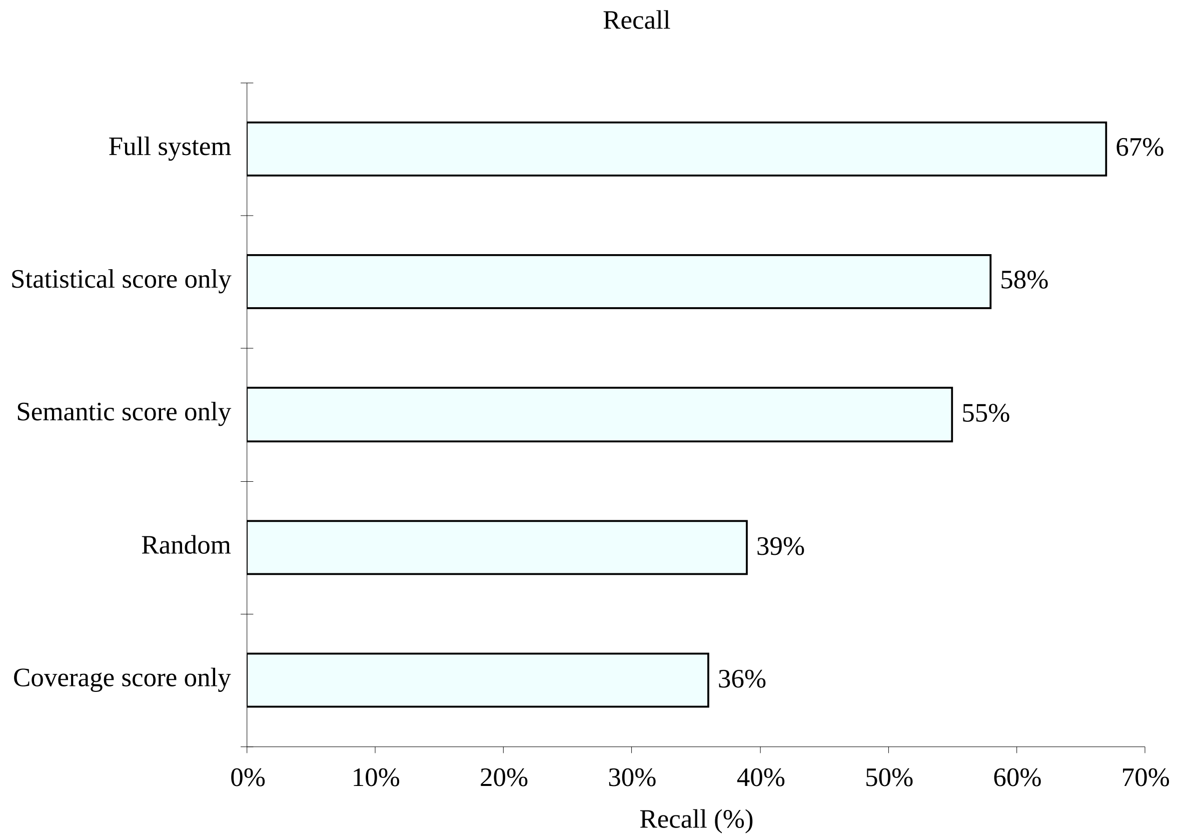


Figure 8: Recall for ablation study

since their combination yields results that are better than either individually. These results confirmed our earlier results with a small corpus of questions, which showed an even more dramatic benefit to the combination of methods.

Figure 9, which shows the recall vs. rejection analysis for these conditions, has more evidence for the difference between the two measures. The curve for the semantic scoring condition is particularly interesting. Although recall in this condition is weaker than the system as a whole, this metric shows good rejection performance. This suggests that the application of semantic information might be used specifically to improve rejection.

Tests for optimal parameters

We ran tests to determine the optimal assignment of weights to the parameters in our matching score formulas. The value of the exact string match was set to 1.0. The possible values for matches between morphological relatives were taken from the set {.5, .65, .85}. The weights for morphological matches were lower than for exact string matches to account for cases of morphological ambiguity. The depth values, the D parameter in the formula, were in the interval [1, 6]. Depth values specify the maximum number of links in WordNet that can be traversed before an intersection is found. The values of the highest WordNet score, the H parameter, were taken from the set {.4, .5}. The value of the lowest WordNet

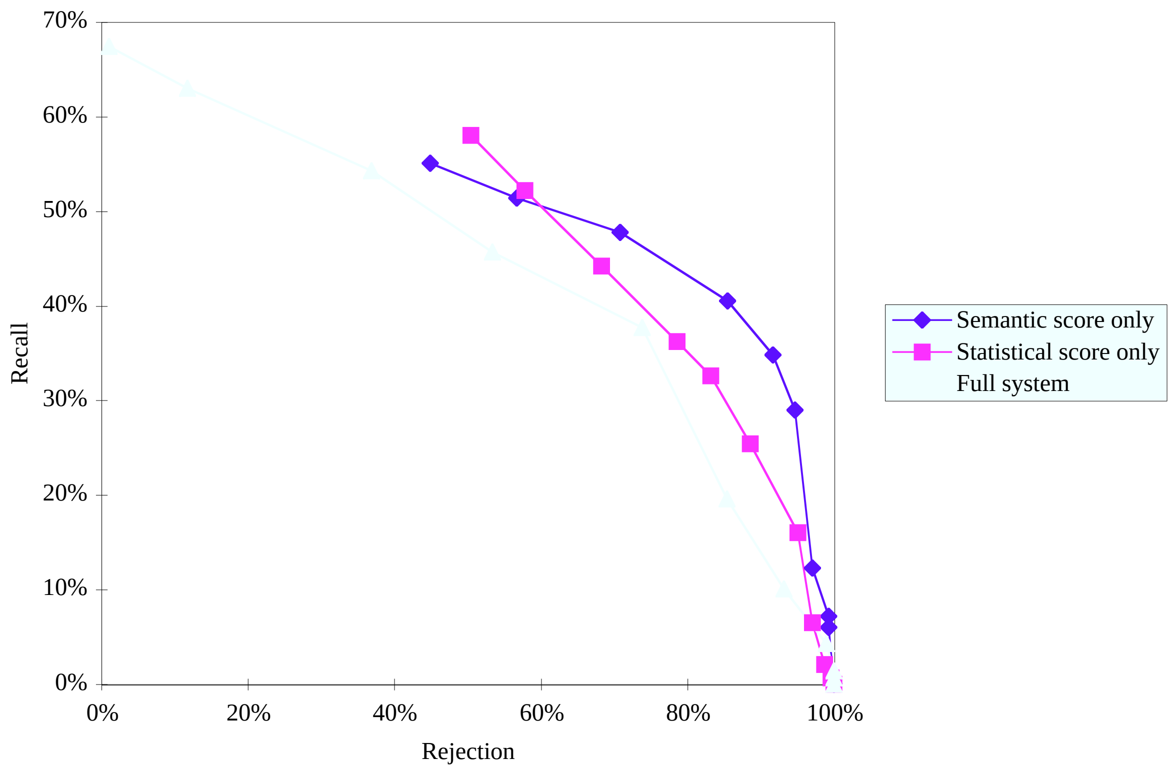


Figure 9: Recall vs. rejection for ablation study

score, the L parameter, was set to 0.0. We have also experimented with two ways to reduce the matrix of WordNet semantic scores. The first method was to compute the average of row maxima. The second method was to return the maximum of row maxima.

We tested 1200 configurations. Each configuration corresponded to one setting of parameters and was tested for recall on a sample of 135 questions. Each of the questions had at least 1 answer in the FAQs known to the system. The table in Figure 10 gives the weights of the parameters for the top 9 configurations that distinguished each configuration from the other 8. The weights of the other parameters were the same in all 9 configurations: the highest WordNet score, the H parameter, was .5, the lowest WordNet score, the L parameter, was 0, the method of reducing the matrix was to average row maxima. All of the 9 configurations achieved a recall of 66%.

There are several interesting observations one can make about these statistics. One observation is that all of the top configurations traversed at most 2 WordNet links. This is in line with the fact that semantically relevant words are likely to be found in the most immediate neighborhood of the original word. Another observation is that all of the top configurations used the average of the row maxima method of matrix reduction. The conclusion we drew from it is that similarity is stronger when one averages several signals from different sources instead of trusting one, albeit high, signal. For example, the questions "What happens when my computer crashes?" and "Where can I buy a good computer?" have a very strong signal on the word "computer" but have little to do with each other otherwise.

W=WordNet	*T=Term Vector*	*Morphology*	*D=Depth*
.6	.4	.65	2
.6	.4	.85	2
.6	.4	.85	1
.2	.8	.5	2
.2	.8	.5	1
.2	.8	.65	2
.2	.8	.65	1
.2	.8	.85	2
.2	.8	.85	1

Figure 10: Different Weights of Parameters in Top 9 Configurations

Vector similarity

The second set of configuration tests was designed and carried out in order to measure relative strengths and weaknesses of our vector similarity measures. Figure 11 summarizes the results of the tests in a table. We have tested 4 vector similarity measures on each of the top 9 configurations. The question set remained unchanged. Each similarity measure was tested both with real and binary weights. We have used the following measures:

$$Cosine(u_i, f_j) = \frac{\sum_{i=1}^{t} u_i f_i}{\sqrt{\sum_i^t u_i^2 \sum_i^t f_i^2}}$$

$$Dice(u_i, f_j) = 2\frac{\sum_i^t u_i f_i}{\sum_i^t u_i + \sum_i^t f_i}$$

$$Tanimoto(u_i, f_j) = \frac{\sum_{i=1}^{t} u_i f_i}{\sum_{i=1}^{t} u_i + \sum_{i=1}^{t} f_i - \sum_{i=1}^{t} u_i f_i}$$

$$Overlap(u_i, f_j) = \frac{\sum_{i=1}^{t} u_i f_i}{min(\sum_i^t u_i, \sum_i^t f_i)}.$$

The table reveals a surprisingly good performance of *binary overlap* and a very consistent performance of *real cosine*. Our explanation for why binary overlap turned up such a good performance is as follows. Most term vectors for questions in our sample set are significantly shorter that the term vectors of Q&A pairs they are matched against. Since the *overlap* measure normalizes by the minimal sum of the weights, the matching scores for similar vectors are higher than when the *cosine* measure is used. We then compared the performance of the *cosine* and *overlap* measures in a separate experiment. Each technique was tested with both binaries and reals. The results showed *real cosine* to be statistically insignificantly better at rejection than *binary cosine*, *binary overlap*, and *real overlap.*

Cosine		Dice		Tanimoto		Overlap	
Real	Binary	Real	Binary	Real	Binary	Real	Binary
.66	.65	.61	.59	.53	.59	.60	.68
.66	.65	.61	.58	.54	.60	.60	.68
.66	.65	.61	.59	.54	.60	.60	.68
.66	.59	.59	.56	.54	.58	.60	.67
.66	.59	.59	.56	.53	.58	.60	.67
.66	.59	.59	.56	.54	.58	.60	.67
.66	.59	.59	.56	.54	.58	.60	.67
.66	.59	.59	.56	.54	.58	.60	.67
.66	.59	.59	.56	.54	.58	.60	.67

Figure 11: Similarity Metrics

Analysis of Failures

We analyzed the cases when FAQ Finder failed to find the correct Q&A pair when it was in the FAQ. We classified failures into three broad categories which we call *inference failures*, *domain terminology failures*, and *term selection failures*. In this section we describe each of these categories and suggest ways to avoid them in the future.

Inference failures

Many questions for which FAQ Finder failed to find the right match require an ability to make inferences. An example of this type of question is "What makes a speaker good?". Putting aside the metaphorical nature of this question for the time being, the correct answer to it is the Q&A pair from the `audio_FAQ` whose question is "What should I listen to when evaluating speakers?" The only term that the two questions have in common is "speaker" but since many other questions in that FAQ contain the same term its *tfidf* value is very low. Nor are there any connections for the marker passing algorithm to find in WordNet between "make", "speaker", and "good" on the one hand, and "listen", "evaluate", and "speaker" on the other. This is not at all surprising once one recalls that WordNet maintains connections only between words of the same part of speech. Thus one can attempt, albeit in vain, to find a connection between "evaluate" and "listen" because both are verbs, but not between "evaluate" and "good" as the former is a verb while the latter an adjective.

What is required for the correct match in this case is an inference chain that says that people evaluate an audio device called speaker by listening to the quality of sound it produces, and if the quality of sound is good, it makes the speaker good too. This inference chain is domain dependent. Since each of the FAQs used in the system constitutes a different domain, one would have to do a great deal of knowledge engineering in order to do the inference-based matching. One can argue that this problem has an incremental solution: as soon as there is an inference model for a given domain, integrate it into the matcher. There is nothing wrong with this idea in theory. In practice, however, there are two concerns. The first concern is efficiency: the inference mechanism, even if it is domain dependent, may not be fast enough for quick web interactions that typical users of FAQ FINDER expect from the system. The second, perhaps more serious, concern is the translation of natural language inputs into the right knowledge representation language that the inference mechanism supports (Norvig, 1987): the natural language understanding problem which we do not seek to tackle head-on.

The real question for this is whether one can approximate inference without building an inference mechanism. We believe that the co-occurrence of terms may give a simple affirmative answer to this question. When people talk about a topic, they tend to use same sets of words over and over again. For instance, when one talks about "baseball", there is a high probability that one will use words like "ball" and "bat." Thus if one knows the words that tend to co-occur with the words of an input question one can use them in one's matching operations. The text of the `audio_FAQ` shows that the words "speaker" and "listen" have a high degree of co-occurrence which can be used in finding the correct match. We are looking for effective heuristics to find co-occurrences and use them in matching operations. One such heuristic would be to augment WordNet's semantic relations with a weaker domain-specific relation like *co-occur*. For example, if one knows that a particular FAQ is about baseball and words "ball" and "bat" show a high degree of co-occurrence, one can add the relation *co-occur("ball", "bat", "baseball")* to WordNet's database of semantic relations.

Domain terminology failures

Another category of failures comprises questions that contain domain specific terminology. For example, the question "What is the best counting system for blackjack?" is answered under the question "What BJ counting system is most effective?" which is not found by the system because of its inability to tell that "BJ" is a FAQ specific abbreviation of "blackjack" coupled with the fact that the terms "counting" and "system" occur in many other questions of the FAQ so that their *tfidf* values are insignificant. If the system knew that the terms "BJ" and "blackjack" were equivalent in the context of the `gambling_FAQ`, it would be able to find the match.

Many FAQ files contain a section where the most common abbreviations are explained. For example, the tagged version of the `gambling_FAQ` has the following section:

```
:QUE
What do these funny acronyms mean ...
:QUE

:ANS
(Michael Hall)

  The acronyms that are often used in blackjack articles in rec.gambling
  are listed below.
  Abbreviations:
BJ  = blackjack
BSE = Basic Strategy Edge
H17 = Hit soft 17 (dealer must hit)
S17 = Stand on any 17 (dealer must stand)
DOA = Double On Any first two cards
D10 = Double on 10 or 11 only
DAS = Double After Splitting is allowed
RSA = Re-Splitting Aces is allowed
ESR = Early Surrender
LSR = Late Surrender
O/U = Over/Under 13 side bets are allowed

:ANS
```

Here ":QUE" and ":ANS" are the tags that FAQ Minder uses to label questions and answers respectively. Although the problem of thesaurus construction has been one of the hardest problems in information retrieval, we believe that the task for acquiring FAQ specific sets of terms is more manageable. As one can see in the above tagged excerpt, FAQ writers typically use easily identifiable patterns and lexical cues to present domain specific terminology. The most common lexical cues are the terms "acronym", "abbreviation", and "terminology." Such lexical cues will help FAQ Minder home in on the terminology section. Once the terminology section is found, a set of regular expressions can be used to find term definitions like "BJ = blackjack."

Term selection failures

The third category of failures is made up of questions whose matches were not found because of the problems with our term selection mechanism. Each Q&A pair is turned into a vector of terms whose weights are standard `tfidf` values. This simple measure was a good starting point to test our original idea that a combination of semantic and statistical techniques works better than any single approach. Sometimes, however, it fails to give high weights to important terms. For example, the question "Can I get AIDS from kissing?" has its answer under the question "How is AIDS transmitted?" The term "kiss" is in the answer but its *tfidf* value is not high enough for the matcher to retrieve the Q&A pair.

Another problem with the term selection method based exclusively on *tfidf* was undue weight given to semantically useless words. For example, a question such as "Where can I find woodworking plans for a futon?" retrieves questions that incorporate the word "woodworking" as strongly as those that contain "futon", even though "futon" should be a much more informative term inside the `woodworking_FAQ` than "woodworking", which applies to everything. The problem is that the term "woodworking" does not appear that often in the FAQ despite its close semantic relation to words that do appear.

There are two ways to address these problem. One way is to divide long answer texts into several chunks and build a term vector for each chunk. At run time the matcher can retrieve a Q&A pair only when a fraction of its answer chunk vectors return sufficiently high scores. This solution would find the right answer to "Can I get AIDS from kissing?" because the answer text of "How is AIDS transmitted?" has a paragraph where the term "kiss" occurs several times. The second way is to use the *tfidf* measure in combination with other term selection techniques. As a term weight measure *tfidf* is based on global word frequency counts. The advent of full text documents has created new possibilities. One can exploit the sequential occurrence pattern of a word over the structural elements of a document to decide whether it bears content or not. Since FAQ files have several structural constraints on their texts, it makes sense to experiment with term selection techniques that are more sensitive to the document structure than *tfidf.*

One technique that seems particularly suitable for our purposes is *condensation clustering* (Bookstein, Klein, & Raita, 1995). The basic idea is to look at the distribution of a term over a set of receptacles, i.e. sentences, paragraphs, Q&A pairs, chapters, etc, and see if the distribution has a statistically significant deviation from the random distribution. If that is the case, the term is likely to be content-bearing. If, for example, the text of the answer to "How is AIDS transmitted?" is broken into paragraphs, the term "kiss" will exhibit a high degree of *condensation clustering.* Experimental results also show that *condensation clustering* is good at detecting randomness in term distributions. Thus it should detect that the term "woodworking", which is semantically useless in `woodworking_FAQ`, is distributed randomly in its Q&A pairs.

Failures of assumptions

Another type of problem commonly encountered with FAQ FINDER is related to violations of the assumptions about FAQ files discussed at the beginning of this paper: Q&A format, locality of information, question relevance, and sufficiency of general knowledge.

We have found some instances in which these assumptions are violated. For example, FAQ writers frequently use headings to mark sections of their documents and rely on the reader's interpretation of those headings in their question writing. In the "Investment FAQ" file, the following text can be found:

```
Subject: Analysis - Technical:
...
   Q: Does it have any chance of working?
...
```

The "it" is of course intended to refer to technical analysis. However, FAQ FINDER is currently not capable of making use of this referent because it lies outside the Q&A pair, making it more difficult to match against a question like "Does technical analysis work?" Part of our intent as we automate the tagging process is to make heading information available to the matcher.

There are other more difficult cases of ellipsis found in FAQ files. In the "Wide-Area Information Server FAQ," the following passage can be found:

```
   Q: What is Z39.50?
```

```
    A: ...
    Q: Do they interoperate?
    A: ...
```

The pronoun "they" refers to both Z39.50, an information retrieval standard, and WAIS, the subject of the FAQ. We do not expect FAQ FINDER to be able to dissect references that are this oblique. It would, however, be useful to refer back to earlier questions if there is no heading information with which to resolve a referent.

One FAQ-specific phenomenon we have encountered is the use of *metasyntactic variables*, meaningless pieces of text that stand in for a filler, which can vary. For example, the "Pool and Billiards FAQ" contains the question

```
    Q: What are the rules for XXX?}
    A: STRAIGHT POOL...
       EQUAL OFFENSE...
       NINE BALL...
```

Metasyntactic variables often have a distinct form and can be easily recognized. We anticipate that a mechanism similar to a heading recognizer could be used to recognize the sub-answers within a multi-part answer such as this. Not every variable can be so treated, however. The "Woodworking FAQ" contains the question

```
  Q: Should I buy a Sears blurfl?
```

The answer does not enumerate the entire catalogue of Sears power tools: the same advice is intended to apply to all. The reader is supposed to be capable of matching the nonsense word against the name of any power tool. This is exactly the type of domain-specific knowledge that we have sought to avoid including in FAQ FINDER. FAQ FINDER can successfully match this question against questions like "Are Sears power drills a good buy?" because the word "Sears" is sufficiently distinctive, but it would fail to match against a question like "What kind of power drill should I buy?"

Future Work

Rejection

One of the most obvious open questions in the research we have done so far is the problem of improving the system's rejection of unanswerable questions. We have concentrated our tuning of the system on maximizing recall, so that questions that are answered in our files will be correctly handled as often as possible. However, it is also useful to be informed that an answer does not exist within a FAQ file. This may suggest to the user that the question should be submitted to the FAQ's related newsgroup. [10]

[10] If rejection were sufficiently good, the system could automatically exercise this option itself.

One way of approaching this problem is to focus on the small retrieved set of Q&A pairs before they are returned to the user. We know from our evaluation that if the answer is present in the FAQ file, the system is likely to find it, therefore if none of the Q&A pairs returned by the system are in fact a good answer, the chances are good that the system should report that the answer does not exist. We also know that semantic information seems to have better rejection characteristics than statistical information. We may therefore be able to perform a more in-depth analysis, possibly involving slightly deeper natural language processing, to accept or reject the returned set of questions. Because this set is small, such intensive processing would not be as computationally prohibitive as performing deeper natural language processing throughout the entire matching process.

Another possibility to improve rejection by improving our term selection mechanism. Our analysis of the experimental results shows that although *tfidf* is a good term selection measure sometimes it gives low weights to semantically important words and high weights to semantically irrelevant words. As a measure of term importance, *tfidf* is based on global counts of word frequencies. We have outlined how term selection can be improved if *tfidf* is used in combination with techniques that are more sensitive to the overall structure of text. One such technique is *condensation clustering*. The basic idea behind this technique is to look at the distribution of a term over a set of receptacles, i.e. sentences, paragraphs, Q&A pairs, chapters, etc, and see if the distribution has a statistically significant deviation from the random distribution. If that is the case, the term is likely to be content-bearing.

File processing

One area of active and continuing research is the automatic extraction of structural elements from the FAQ files. We are working to improve the system's performance on extracting Q&A pairs, but we would also like to extend its reach to extract sectional headings and metasyntactic variables. Other specific elements present in many FAQ files may also be worth attending to: for example, lists of addresses and lists of references. The word-for-word content of such sections match poorly with questions like "What books are there on antique radios?" – the word book might not even appear in a list of titles and authors, but if the section of the FAQ could be reliably identified as a bibliography, the possibility of such matches would be greatly enhanced (Kulyukin, Hammond & Burke, 1996).

We are investigating ways to improve the system's response time. A significant component of the matching time is the input of FAQ file-specific auxiliary files. There are two mechanisms that may improve our handling of these files. One possibility is caching: we can store auxiliary information for some limited number of FAQ files in the Common Lisp image, purging on a least-recently used basis. This technique would only be successful if the usage pattern of the system is such that files are likely to be reused. For example, a user can ask several related questions in sequence. Only empirical experience with the system as a public utility will reveal whether or not this is the case, and what caching parameters might be appropriate.

Augmenting semantic knowledge

As was noted earlier, one common question-answering failure of the system has its origin in its inability to do inference. We have argued that although in theory inference can improve the question-answering capabilities of FAQ FINDER it is not an attractive practical solution. We believe that in many cases one can approximate inference with statistical techniques based on co-occurrences of terms. One direction of future research will be aimed at finding such co-occurrences and adding them to WordNet's database of semantic relations.

An important part of maintaining the performance of FAQ FINDER on a large set of FAQ files will be the incorporation of new vocabulary items into WordNet. Since WordNet was formed from a corpus of everyday English, its vocabulary does not include many technical terms or proper nouns. Unfortunately, due to the technical nature of many FAQ files, technical terms and proper nouns constitute a significant portion of the domain vocabulary of these files. In addition, these terms can be the most useful in retrieving relevant Q&A pairs, since they are often the most specific and discriminating terms. Thus, the fact that they are missing from WordNet can significantly impair the performance of FAQ FINDER. Although the thesaurus construction problem remains open in information retrieval, we believe that for our purposes an good approximate solution to this problem can be based on recognizing certain regular expressions and lexical cues in the texts of FAQ files.

We are also investigating ways in which information obtained from the parses of questions can be used to automatically acquire additional terms and build the appropriate synonym and hypernym links for these terms in one of the WordNet hierarchies. We rely on feedback from the user to tell the system when a good match has been found between a user question and a Q&A pair.[11] If the user indicates the system retrieved a relevant answer, then any words in either the user or the FAQ question that are not contained in WordNet have the potential to be acquired. The system attempts to match the unknown word with a synonym in the other question. Both questions are parsed, and position in the parse tree is used to determine which words are candidate synonyms of the unknown word.

Consider the following example. Say the user asks the question, "How can I recover an unsaved Word file if my Macintosh crashes?" Let us assume that "Macintosh" is not encoded in WordNet. If FAQ FINDER retrieves a question like "Is it possible to recover unsaved files when my computer crashes," and the user indicates that this is a good match, then the parse trees of the user and FAQ questions can indicate that "Macintosh" and "computer" are likely to be synonyms, since both appear as the subject of the verb "crashes". This suggests that "Macintosh" should be entered in WordNet as a synonym or hyponym of "computer".

Since the matching process between question pairs is likely to incorrectly propose some synonyms of unknown words, our approach is to collect synonyms for unknown words over time, and propose new WordNet entries by analyzing collections of possible synonyms for each unknown term. Clustering algorithms are likely to be of use here in determining the likely best entries for an unknown word. Proposed synonyms which are "outliers"

[11] Using our previous local version of FAQ FINDER, about 20% of users gave this type of feedback. We expect that the improved interface of FAQ FINDER 2.0 will increase the frequency of user feedback on performance of the system.

from a cluster of other proposed synonyms could be discarded as probably incorrect, based on a clustering analysis. In addition, ambiguous unknown words could be detected by finding more than one cluster of synonyms. For example, for the word "Macintosh", our system might collect a list of synonyms which include terms such as "computer," "pc," "workstation," "apple," and "fruit." This would suggest two clusters, corresponding to the two possible meanings of "Macintosh."

Augmenting the FAQ file corpus

At over 2000 files, the `news.answers` collection is reasonably large, but it is a small subset of the set of FAQ files that are obtainable over the Internet. Many newsgroups and mailing lists maintain their FAQ files at FTP sites other than the MIT one, and many individuals, groups, and organization are creating FAQ files resident on the WWW. We see these alternate sites as an excellent source for additional files to expand the score of FAQ FINDER. There are several research issues to be addressed to make such incorporation possible.

- The tagging mechanism would have to be reworked to handle HTML files. This would likely be easier than handling straight text files, since the HTML structuring tags provide clues that are much less ambiguous than the text structure information we currently use.
- We would need to devise a FAQ file recognition mechanism for use either with our own web crawler or against the results returned by a search engine.
- We would need additional maintenance tools – active mechanisms for staying up-to-date by revisiting web sites.

Finally, a wider FAQ collection will contain more spurious information which may degrade the usefulness of FAQ FINDER. We are developing heuristics that will help the system do some independent information filtering.

Answering questions for an organization

Organizations have to answer questions. Clients who have questions for an organization often have to spend a long time navigating its structure before they get answers. The organization's experts often feel overwhelmed by having to field the same questions time and time again or to route new questions to colleagues whose expertise is more appropriate. As the body of expertise in the organization grows, the need to satisfy clients and to reduce the burden on its experts becomes urgent.

If used uniformly across an organization, a system similar to FAQ FINDER can help an organization answer clients' questions efficiently and reduce the burden on its experts. We are investigating this type of technology in a system called the Information Exchange. The Information Exchange is based on the assumption that the organization's expertise can be structured as a hierarchy of topics. For example, the Information Exchange of a university may have the following topics at the top of its hierarchy: "Admissions", "Academic Units", "Alumni", "Campus Life and Services." To answer questions on a topic, any individual in the organization can register as an expert on it. An expert's knowledge of a topic is organized as a FAQ file. The Information Exchange answers client's natural language questions by first finding relevant experts and then searching their FAQs. Since each FAQ file is associated with an individual, questions that are not answered within the file can be routed to that individual via email for answering and eventual addition to the personal FAQ itself. We are developing a system called Q&A that aims to build the tools required to simplify the creation and maintenance of such personal FAQ files.

We envision a system in which a large organization might maintain such files across its information servers. For example, a student might ask the University of Chicago Information Exchange, "What are the prerequisites of CS 219?" The system would use multiple FAQ file collections to determine what personal FAQ file is best suited to answering this question, in particular, that of a departmental assistant in the Computer Science Department. It would then perform a FAQ Finder-like question-matching step to find the answer within that person's personal FAQ file.

Answers from other types of semi-structured text

As discussed above, RTFM archive contains many files that are not in question-answer format, yet are structured and contain the answers to questions. One particular class of texts that may be interesting to explore is the class of ad-hoc textual tables. For example, Figure 12 shows a section from the `Star-Trek_locations` file.

We are looking at a two-dimensional table: location types on one dimension (here, star systems), proper name on a second dimension, with an entry that is a list of episodes, with an optional descriptive tag in parentheses.[12] Such tables can be thought of as a representation of a simple database structure. The task of the "tagger" in this case would be to attempt to derive a loose meta-level description of the contents of the database. If that underlying database could be recovered, it would be possible to answer questions such as "What Star Trek episodes included the Minos Korva system?" or "What star system does the *Bread and Circuses* episode occur in?"

A somewhat less ambitious approach to these files is to use passage-based retrieval techniques (Callan, 1994). A document can be split into chunks of text.[13] Each chunk is treated as a separate document. An answer to the user query, in this case, is a place in the FAQ that is likely to contain relevant information.

Conclusion

We have described FAQ Finder, a functioning knowledge-based information retrieval system, that relies on the knowledge engineering inherent in FAQ files distributed on the Internet. The FAQ Finder project has shown that when there is an existing collection of questions and answers, as found in the FAQ files, question answering can be reduced

[12]Non-frivolous examples of such files also exist in the RTFM archive.

[13]The TextTiling technique proposed in (Hearst, 1993) seems particularly suitable here.

```
------------------------------------------------------------

                    SOLAR/STAR SYSTEMS

------------------------------------------------------------

Acamar
       TNG "The Vengeance Factor"
Alpha Centauri
       TOS "Metamorphosis"
...
Minos Korva
       (11 lightyears from McAlister C5 Nebula)
       TNG "Chain of Command, Part II"
       (site of a Cardassian incident)
       TNG "Gambit, Part I"
Mira (six planets)
       TNG "Conspiracy"
```

Figure 12: A section from the `Star-Trek_locations` file.

to matching new questions against Q&A pairs. This is a considerably more tractable task than question understanding. The system combines statistical measures and shallow lexical semantics in matching users' questions against Q&A pairs recorded in FAQ files. Our evaluations, although conducted with a small corpus of questions, demonstrate the effectiveness of the system.

The power of our approach rises from the fact that we are using knowledge sources that have already been designed to "answer" the commonly asked questions in a domain and as such are more highly organized than free text. We do not need our systems to actually comprehend the queries they receive (Lang, et al. 1992) or to generate new text that explain the answer (Souther, et al. 1989). They only have to identify the files that are relevant to the query and then match against the segments of text that are used to organize the files themselves (e.g., questions, section headings, key words, etc.).

We have argued that although in theory inference can improve the question-answering capabilities of FAQ FINDER it is not an attractive practical solution. We believe that in many cases one can approximate inference with statistical techniques based on co-occurrences of terms.

Ultimately, FAQ files are a social phenomenon, created by people to record and make public their understanding of a field. In general, the FAQ FINDER project is interesting in that it uses not just the existing archives on the Internet, but also the existing sociology. One of the more powerful aspects of the newsgroups is the collective desire on the part of the users to "get it right." This drive has already resulted in the existence FAQ files themselves. Our aim in FAQ FINDER is to further this goal by making the answers recorded in FAQ files more widely available.

Acknowledgments

In addition to the authors, the development team for FAQ FINDER 2.0 consisted of Jay Budzik and Benjamin Young of the University of Chicago; and David Mortman of Commonwealth Edison. Previous versions of the system benefited from contributions by Edwin Cooper, Xiaobin Fu, Julia Kozlovsky, Charles Martin, and Tom McDougal. Research on FAQ FINDER has been supported with funds from the Office of Naval Research and with hardware donations from Apple Computer.

References

Bookstein, A., Klein, S., Raita, T. 1995. Detecting Content-Bearing Words by Serial Clustering - Extended Abstract. In *ACM SIGIR 1995*, pp. 319-327.

Buckley, C. 1985. Implementation of the SMART Information Retrieval Retrieval [sic] System. Technical Report 85-686, Cornell University.

Burke, R., Hammond, K., & Cooper, E. 1996. Knowledge-based information retrieval from semi-structured text. In *AAAI Workshop on Internet-based Information Systems*, pp. 9-15. AAAI.

Callan, J. P. 1994. Passage-Level Evidence in Document Retrieval. In *ACM SIGIR 1994*, pp. 302-309.

Collins, A. M. and Quillian, M. R. 1972. How to Make a Language User. In E. Tulving and W. Donaldson, *Organization of Memory.* New York: Academic Press.

Cutting, D., Kupiec, J., Pederson, J., & Sibun, P. 1992. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing.* ACL.

Grady Ward, 1993. *Moby Part-of-Speech II.* Computer file. Arcata, CA: Grady Ward.

Hearst, M. A. 1993. TextTiling: A quantitative approach to discourse segmentation. Technical Report Sequoia 93/94, Computer Science Department, University of California, Berkeley.

Hornby, A. S. 1984. *Oxford Student's Dictionary of Current English.* Oxford, UK: Oxford University Press.

Kolodner, J. 1993. *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann.

Kulyukin, V., Hammond, K. & Burke, R. 1996. Automated analysis of structured on-line documents. In *AAAI Workshop on Internet-based Information Systems*, pp. 78-86. AAAI, 1996.

Lang, K. L.; Graesser, A. C.; Dumais, S. T. and Kilman, D. 1992. Question Asking in Human-Computer Interfaces. In T. Lauer, E. Peacock and A. C. Graesser *Questions and Information Systems* (pp. 131-165). Hillsdale, NJ: Lawrence Erlbaum Assoc.

Lenhert, W. 1978. *The Process of Question Answering.* Hillsdale, NJ: Lawrence Erlbaum Assoc.

Miller, G. A. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11).

Norvig, P. 1987. *Unified Theory of Inference for Text Understanding.* Technical Report No. UCB/CSD 87/339, University of California at Berkeley.

Ogden, W. C. 1988. Using natural language interfaces. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 205-235). New York: North-Holland.

Quillian, M. R. 1968. Semantic Memory. In *Semantic Information Processing*, Marvin Minsky, ed., pp. 216-270. Cambridge, MA: MIT Press.

Salton, G., & McGill, M. 1983. *Introduction to modern information retrieval.* New York: McGraw-Hill.

Souther, A.; Acker, L.; Lester, J. and Porter, B. 1989. Using view types to generate explanations in intelligent tutoring systems. In *Proceedings of the Eleventh Annual conference of the Cognitive Science Society*, pp. 123-130. Hillsdale, NJ: Lawrence Erlbaum Assoc.