

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

Robin D. Burke, Kristian J. Hammond, & Vladimir 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

December 20, 1996

Preliminary draft – not for citation

Abstract

This paper 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 considerations that have entered into the system and various experiments that influence the system's current implementation. We include results from an evaluation of the system's performance against a corpus of user questions, and show that a combination of semantic and statistical techniques works better than any single approach.

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 can be found in the wide assortment of frequently-asked question (FAQ) files, many 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 about 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 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 should be able to return the appropriate answer.

FAQ FINDER is built on four assumptions about FAQ files:

QA format: All of the information in a FAQ file is organized in question/answer format.

Locality of information: All of the information needed to determine the relevance of a question/answer pair can be found within that question/answer pair.

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

General knowledge: Broad general knowledge 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 question/answer 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. but, 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 question and answer (QA) 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 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 conferences² and described in workshop publications (Burke, Hammond & Cooper, 1996). Local use of the system also enabled us to gather a better test corpus. In

¹`<URL:ftp://rtfm.mit.edu/>`

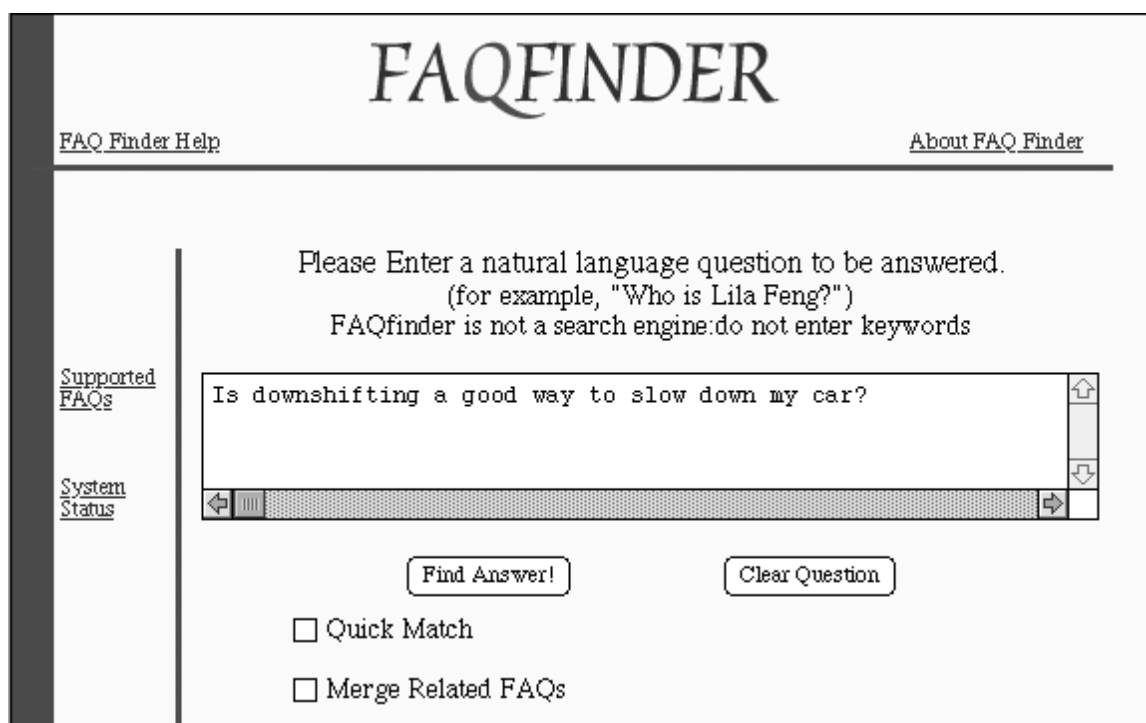
²AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval (MIT, 1995), The 5th International World Wide Web Conference (Paris, 1996), AAAI Workshop on Internet-based Information Systems (Portland, 1996)

the Fall of 1996, we undertook to reimplement the system in 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?



The screenshot shows the FAQFINDER web interface. At the top, the title "FAQFINDER" is displayed in a large, stylized font. Below the title, there are two links: "FAQ Finder Help" and "About FAQ Finder". The main content area contains the instruction: "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". A text input field contains the question: "Is downshifting a good way to slow down my car?". Below the input field, there are two buttons: "Find Answer!" and "Clear Question". At the bottom, there are two checkboxes: "Quick Match" and "Merge Related FAQs". On the left side, there is a vertical menu with links: "Supported FAQs" and "System Status".

Figure 1: Asking a question of FAQ FINDER.

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 his or her question against all three.

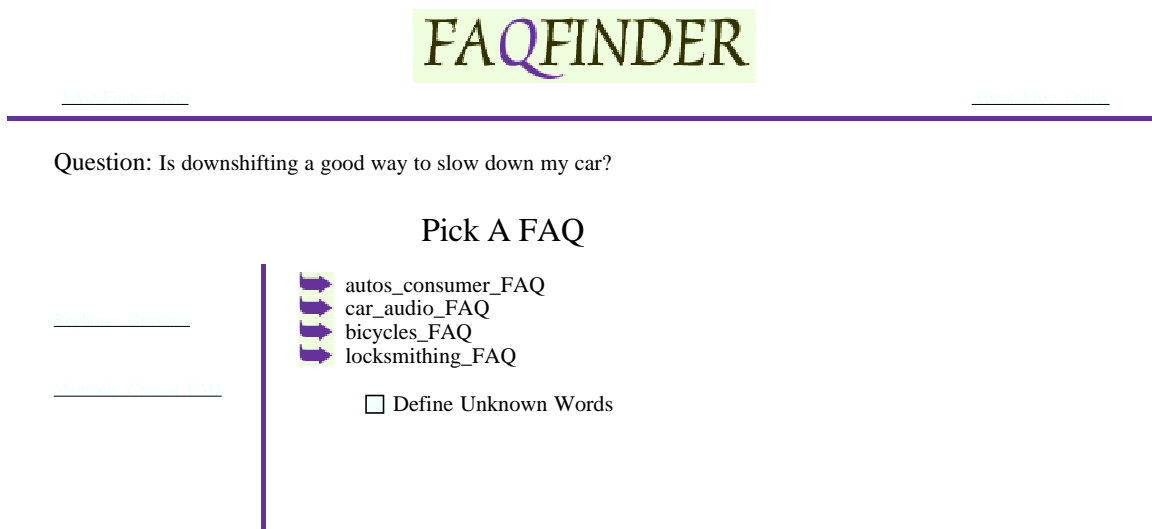


Figure 2: Choosing a FAQ file to match against.

When a FAQ file is chosen (whether by the system or the user), the system iterates through the QA 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 paper has four parts. First, we give a general outline of the FAQ FINDER implementation. Then we will discuss some of our early experiments with the architecture and the results that have informed our present implementation. The next section delves more deeply into the details of the current FAQ FINDER implementation, touching particularly on the issues raised by the World-Wide Web as a platform. Evaluation results follow, including a discussion of our evaluation methodology and metrics. We conclude with a description of on-going and future work on the system.

Technical Overview

As the example shows, interaction with FAQ FINDER occurs in a series of stages. The first step is to narrow the search to a small set of FAQ files likely to contain an answer to the user’s question. The user may or may not need to confirm this choice. Second, each QA pair is matched against the user’s question to find the ones that best match it.

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

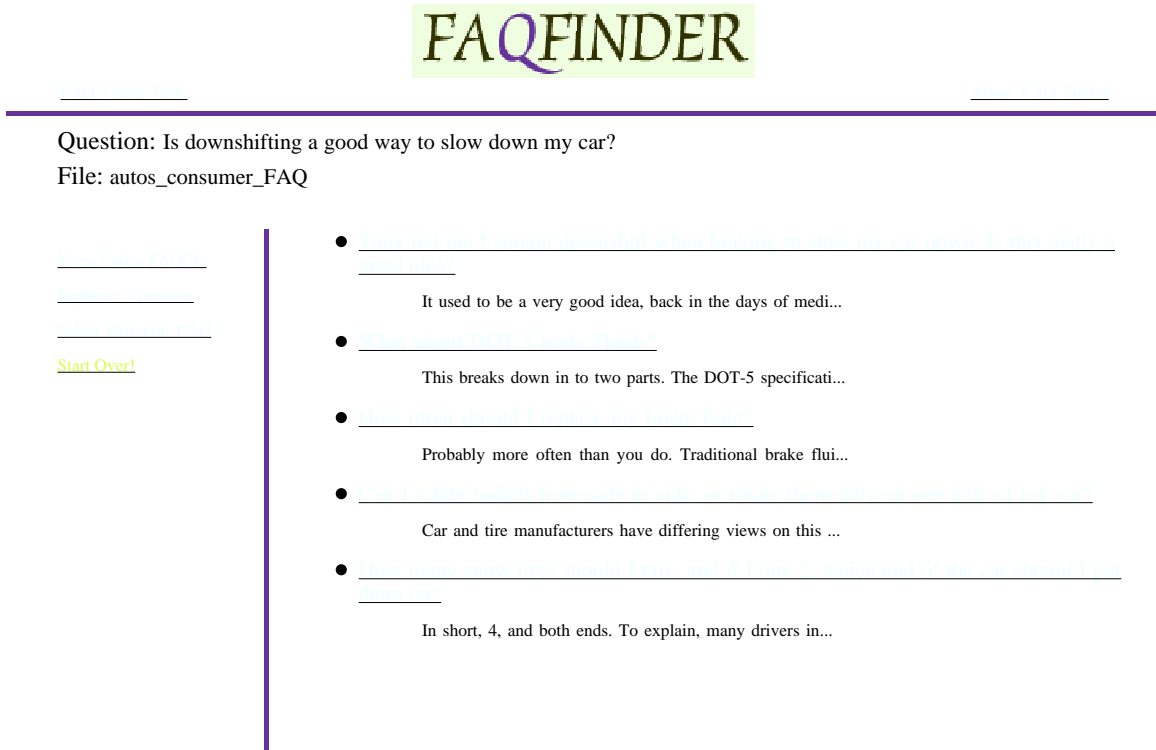


Figure 3: Choosing an answer.

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. The top-ranked files from this procedure are returned to the user for selection, if the user wishes to provide the system with feedback at this point; or are directly passed to the second stage of processing. We have not found it necessary to tinker with the default configuration of SMART. We 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 several metrics in combination to arrive at a score for each question/answer pair: a statistical similarity score, a semantic similarity score, and a coverage score, the percent of words in the user’s question that the question from the file covers.

Statistical similarity

The statistical similarity score at the QA pair level is computed in a manner quite similar to SMART’s document matching. A QA pair is represented by a term vector, a sparse vector that associates a significance value with each term in the QA 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 term frequency (the number of times that a term appears in a QA pair), m is the number of QA pairs that the term appears in the file, and M is the number

FAQFINDER

[FAQ Finder Help](#)

[About FAQ Finder](#)

[Previous Question](#)

[Next Question](#)

[View Other FAQs](#)

[Refine Question](#)

[Start Over!](#)

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.

of QA 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 as a factor to weight the frequency of that term in a particular document. A term that appears in every QA 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 QA 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 vector representing the user's question and the vector representing the QA 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 question/answer pair, taking into account the frequency of occurrence of different terms within the file as a whole. This metric does not require any 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 reasonably 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.

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 question-answer pairs in each file. The semantic matching algorithm in FAQ FINDER is designed to handle variations in lexical content 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?

Here, the difficulty 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?

Here, 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.

Shallow lexical semantics in WordNet

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” can be judged to be 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. Because there is no checking to make sure that complex semantic or syntactic relations are satisfied and we have a fixed length of marker propagation, this marker-passing phase is polynomial in the branching factor of the WordNet network.

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. We have designed FAQ FINDER as a modular system so that many scoring functions can be tested. At the moment, we are using a scoring technique that involves building a matrix of scores and then reducing that matrix to a single number.

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 i th word of the user’s question. Let f_j be the j th 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 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. Words that are identical or a morphological variants are given fixed scores.

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

$$w(u, f) = \frac{\sum_{j=1}^m \forall u_i \max(s(u_i, f_j))}{m}$$

We compute a third measure, c , the degree of coverage of the FAQ question. The intuition behind this measure is to penalize questions that are lacking corresponding words for each word in the user’s question. In other words, we do not care if the FAQ file question answers many questions at once, but we want to make sure that the important concepts in the user’s question are covered. The coverage value is the percent of words in u that have a non-zero s for some f_j .

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

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

where T , W , and C are constant weights associated with term vector, WordNet and coverage scores, respectively. Adjusting these weights adjusts the reliance of the system on each component of the match score.

File management

FAQ FINDER matching algorithm compares user questions against QA pairs from the FAQ file. To identify the QA 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 question/answer 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`, following line was tagged as a question:

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

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?” This line matches with the pattern `^Subject:\s+[\d+\.] +\s+.*\n+$` (this can be read as “start with **Subject**: followed by one or more spaces, followed by digits or periods, followed by one or more spaces, include any arbitrary string, and then end with one or more newline characters). In other FAQ files, this pattern matched the question correctly.

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`, 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 QA 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” package³ 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.

³<URL:ftp://src.doc.ic.ac.uk/computing/archiving/mirror/>

Many of the files in the `news.answers` are not in question/answer 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`.

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 question/answer 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 question/answer 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 QA format or not, but in many cases, manual examination will still be required.

Initial Experiments

Before the creation of the current version of FAQ FINDER, the system went through many different incarnations as we tested matching techniques. The most important areas of experimentation were our experiments with managing the marker passing process and our attempts to make use of abstract question-type as an input to the matching process.

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 is 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, a wildly 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 indicated that the desired goal of improving semantic matching using WordNet 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 are always tagged according to their default (most frequent) word sense. Thus, parsing could, in theory, improve system performance by correctly tagging words whenever it correctly tags a word with its non-default syntactic category, and additionally by correctly identifying which WordNet tree to search for lexical items which are not in the on-line dictionaries. Although tagging does not cause the system to match a FAQ question that otherwise would not be, recall could be improved if rejecting a FAQ question resulted in the correct question/answer pair being promoted to the “top five” list returned to the user. Actual analysis, however, indicates that parsing currently did not have a significant effect.

There are several possible reasons for why this is the case. First, although the parser found a parse for 80% of user questions, it is possible that the parser sometimes finds the wrong parse, and on those occasions mistags words. If this were happening, however, we would expect parsing to actually decrease performance, rather than having no significant effect. A more likely explanation is that default word sense tagging of user questions turns out to be highly accurate; thus, parsing is not significantly improving the tagging of lexical items. Because parsing is computationally expensive and did not seem to be a significant contributor to the system’s performance, our current version of the program simply uses the default part-of-speech information in the on-line dictionary for disambiguation.

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 lexical 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.

What became immediately obvious from our implementation, however, was that syntactic information was insufficient to determine abstract question type 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. The conclusion of our evaluation was that assessment of question type by purely syntactic means did not contribute to successful matching. 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.

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 a stateless protocol. 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 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 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-HTTP⁵ running under Allegro Common Lisp. Multi-processing is handled within 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 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.

⁴Other 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 by as wide an audience as possible, and so are avoiding the cookie mechanism where possible.

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

Preamalyzing 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 question/answer 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 obviously also must read the question/answer pairs in the FAQ file to perform its comparison. As a result, the matching process is inevitably I/O intensive. It is this aspect of the system that we have put the most effort into optimizing.

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 question-asking 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 question/answer pair, called a “question text” or “qtext” representation. The qtext representation for a question/answer 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 QA 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 QA 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 QA 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 transformations, 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 QA pair in each FAQ file and store them in an associated file, for matching against user questions. The user’s question obviously 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 QA 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 more efficient than using the WordNet executables. 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.

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 "table saw" and "power tool" for example. The words themselves might match to a certain extent "saw" and "tool" for example, but the phrase provides much 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.

Evaluating FAQ Finder

Methodology

We have evaluated the performance of FAQ FINDER on a corpus of questions drawn from the log files of the system's use during the period May to December of 1996.⁶ A total of 241 test questions were used to perform the evaluation. 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

⁶Our 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 system in use. The questions submitted by email were longer, more syntactically complex, and more difficult to answer than the questions typically submitted from the web page. Performance on email questions was about 5-10% worse overall.

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, all the answers are probably 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, then, we calculate our version of recall, which amounts to the percent of questions for which FAQ FINDER returns a correct answer when one exists. Our calculation is slightly 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 only partial success. If the user’s question is answered, it is irrelevant that there was another QA 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.

Results

SMART is highly effective at the file retrieval task. 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 5 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

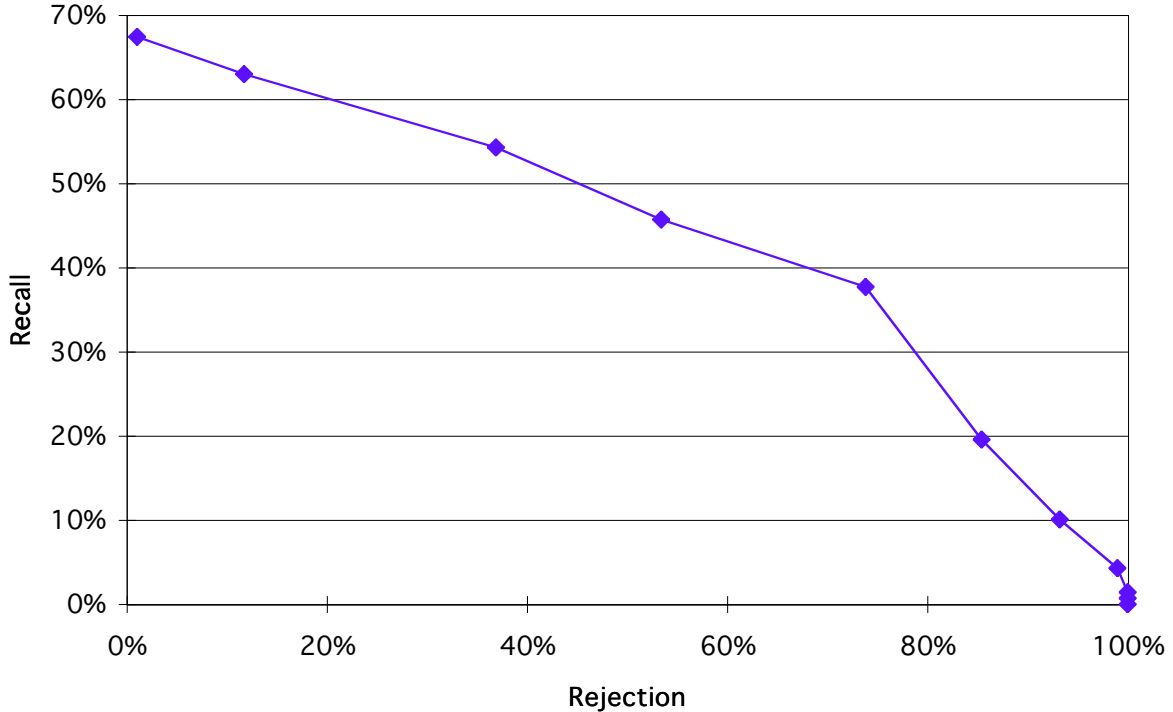


Figure 5: Recall vs. rejection for FAQ FINDER

corpus of questions. There were four conditions: a *random* condition, in which QA 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 6 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. Semantic scoring and statistical scoring had very similar average recall, but are clearly equivalent measures, 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 7, which shows the recall vs. rejection analysis for these conditions, has even more evidence for the difference between the two measures. The curve for the semantic scoring condition is particularly striking. 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.

Discussion

A somewhat unscientific analysis of the failure cases, questions for which the system failed to find answers, suggested that the biggest culprit was usually undue weight given to semantically useless words. For example, a question such as “where can I find woodworking plans for a futon?” retrieves

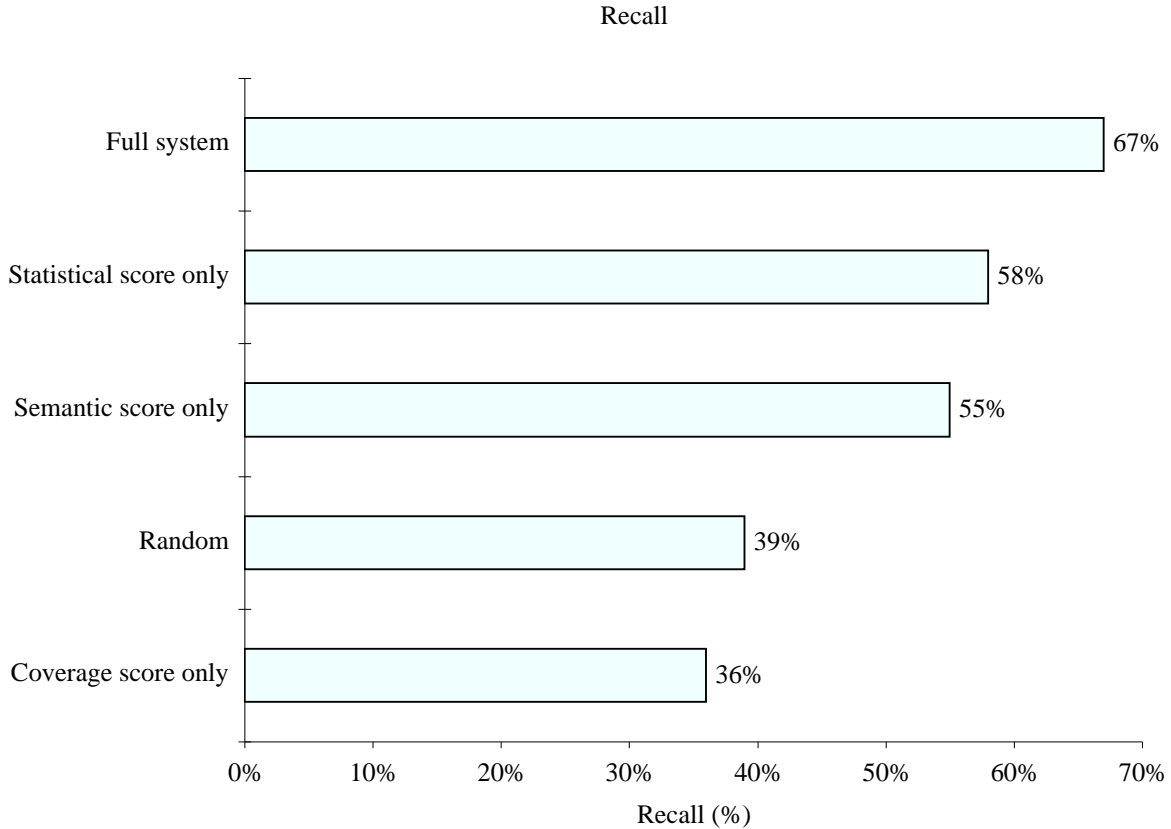


Figure 6: Recall for ablation study

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.

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: question/answer format, locality of information, question relevance, and sufficiency of general knowledge.

Unsurprisingly, we have found many 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 question/answer 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.

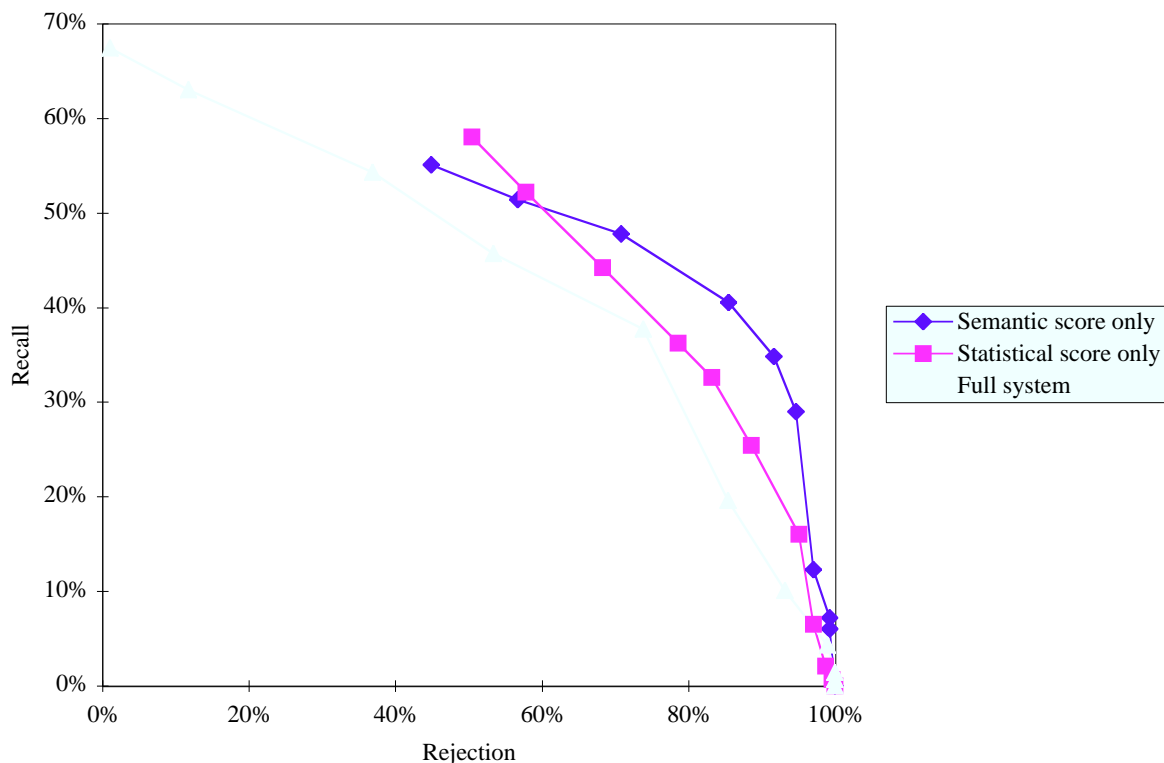


Figure 7: Recall vs. rejection for ablation study

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 reference “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 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

The previous discussion suggest many areas within the scope of the FAQ FINDER project that deserve future research attention. 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 to maximize 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.⁷

One way of approaching this problem is to focus on the small retrieved set of QA 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 QA 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 by definition small, such intensive processing would not be as computationally prohibitive as performing deeper natural language processing throughout the entire matching process.

One area of active and continuing research in the automatic extraction of structural elements from the FAQ files. We are working to improve the system’s performance on extracting question/answer pairs, but we would also like to extend its reach to extract sectional headings and metasyntactic variables also. Other specific elements present in many FAQ files may also be worth attending to specifically: 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 identified as a bibliography, the possibility of such matches would be greatly enhanced.

We are still 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

⁷If rejection were sufficiently good, we could incorporate such an option into the system itself.

information for some limited number of FAQ files in the 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 by a user asking several related questions in sequence. Only empirical experience with the program as a public utility will reveal whether or not this is the case, and what caching parameters might be appropriate.

Augmenting WordNet with machine learning

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 FAQ question/answer 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.

We are 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 FAQ question/answer pair.⁸ 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 question such as “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 entry(ies) for an unknown word. Proposed synonyms which are “outliers” 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.”

⁸Using 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.

Augmenting the FAQ file corpus

At over 2000 files, the `news.answer` 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.

Personal FAQ files for organization question-answering

The FAQ file was created by groups to optimize the effort put into answer questions. With appropriate tools, the same optimization can be achieved for individuals with expertise. If used uniformly across an organization, a FAQ FINDER system can given users the experience of asking a question of an organization and getting a response from particular expert. We are investigating this type of application in a system called Q&A. In this system, individuals develop personal FAQ files, which are treated like the USENET FAQ files in FAQ FINDER. We envision a system in which a large organization might maintain such files across its information servers. For example, a student might ask the general University of Chicago Q&A system, “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 file.

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 file itself. The Q&A project aims to build the tools required to simplify the creation and maintenance of such personal FAQ files so that the benefit of having questions answered automatically outweighs the overhead of using the system.

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 8 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

```

-----
                                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 8: A section from the `Star-Trek_locations` file.

descriptive tag in parentheses.⁹ 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?”

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 to matching new questions against question/answer 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 question/answer pair recorded in FAQ files. Our evaluations, although conducted with a small corpus of questions, have demonstrated 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

⁹Non-frivolous examples of such files also exist in the RTFM archive.

against the segments of text that are used to organize the files themselves (e.g., questions, section headings, key words, etc.).

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

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