RJ 10235 (A0203-003) March 5, 2002
Computer Science

IBM Research Report

Discovering Frequently Asked Questions

Scott Spangler
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Leo Garcia
IBM Global Services
One Alhambra Plaza
Coral Gables, FL 33134

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Discovering Frequently Asked Questions

Scott Spangler
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120
408-927-2887
email: spangles@us.ibm.com

Leo Garcia
IBM Global Services
One Alhambra Plaza, Coral Gables, FL 33134
305-442-3925
email: leog@us.ibm.com

ABSTRACT

Most helpdesk centers document each call to the helpdesk with a short concise text description of each customer's problem and how it was solved. Because of its unstructured nature, the aggregation of this data is difficult to analyze in a meaningful way. In particular we would like to use the unstructured text to answer the question, "What are the current Frequently Asked Questions for a given helpdesk?"

In this paper, we describe an implementation of an algorithm and methodology for discovering Frequently Asked Questions (FAQ's). We utilize text clustering on problem ticket text to determine a set of problem categories. We then use a novel search strategy to find groups problem tickets containing a sufficiently high number of common keywords. We present to the user the most frequently occurring problem ticket groups, with appropriate and readable names for each group. Our claim is that this search strategy serves as a useful method for quickly and automatically determining what the FAQ's are for a given helpdesk. We have validated our approach on many different helpdesk datasets. In particular we compared the FAQ's generated using our approach to those that were generated manually through reading the tickets, and found that our method compares quite favorably to human expert opinion. Finally we describe how the IBM Virtual Helpdesk uses our tool to keep customer helpdesk websites current.

Categories and Subject Descriptors

H.2.8 [Information Systems]: Data Mining

H.3.3 [Information Search & Retrieval]: Clustering

I.5.3 [Clustering]: Similarity Measures.

General Terms

Algorithms

Keywords

Text Mining, Clustering, FAQ

1. INTRODUCTION

In this paper, we focus on analyzing text produced by service and support helpdesks and corresponding knowledge bases. Human helpdesk operation is very labor intensive and therefore expensive. Consequently, automation of helpdesk problem solving represents a key objective for providers of electronic customer services. The first step in automation of a task is to understand it, and one of the first steps in understanding is to segment examples into meaningful categories [1]. The thesis of this paper is that there is much value derived from using our algorithms and methodology to automatically leverage the text within problem tickets to identify areas of frequent recurrence. Our approach uses the analysis, mining, and summarization of problem tickets text using an automated unsupervised clustering algorithm. Our algorithms then automatically discover coherent groups of problem tickets that are sufficiently similar to warrant being called a "Frequently Asked Question (FAQ). The goal of this method is to find the FAQ's at a given helpdesk automatically, with only minimal user input.

Before proceeding further it is necessary to define a little more precisely what we mean by "FAQ". Ultimately which set of examples constitutes an FAQ is a somewhat subjective decision. For instance, it may be that for one helpdesk the set of all "password reset" problems is an FAQ, while at another helpdesk this set is broken down into "windows password reset" and "lotus password reset". The level of detail required will also in turn affect the frequency of occurrence, more specific problem descriptions occurring with less frequency than more general ones. It is important therefore to have a consistent level of generality (within class similarity) across all FAQ's, while at the same time allowing the level of generality to vary depending on the specific needs of the helpdesk. This fact is what makes k-means text clustering (for example) by itself insufficient to find FAQ's. No matter how you specify k (the number of clusters) you still may still end up with a wide range of within class similarity across different clusters.

For our purposes then, we define an FAQ in the following precise and simple way:

FAQ – a set of at least N examples taken from the same text cluster all sharing the occurrence of at least M keywords.

The correct level of generality is provided by the user adjustable parameter M, and the total number of FAQ's generated can be adjusted through the parameter N. The criteria of all examples in an FAQ residing in the same text cluster serves two purposes. First it improves the quality of the FAQ by insuring that all examples within it do not merely share a set of unrelated (and possibly irrelevant) words, but also all have at least some minimal degree of similar content overall. Second it greatly reduces the search space of possible example groupings to choose from, making the problem of finding the correct FAQ's tractable even for large helpdesk problem sets.

In this paper, we will illustrate our results and methodology on a text data set containing over 10,000 helpdesk examples obtained from an IBM Global Services support center. A typical text document (known as a *problem ticket*) from this data set consists of a brief description by the helpdesk agent of the exchanges between an end user and an expert helpdesk advisor, for example:

```
1836853 User calling in with
WORD BASIC error when opening
files in word. Had user
delete NORMAL.DOT and had her
reenter Word, she was fine at
that point. 00:04:17 ducar
May 2:07:05:656PM
```

Problem tickets may be comprised only of a single symptom and resolution pair as in the above example, or they may span multiple questions, symptoms, answers, attempted fixes, and resolutions--all pertaining to the same basic issue. Problem tickets are *opened* when the user makes the first call to the helpdesk and *closed* when all user problems documented in the first call are finally resolved in some way. Helpdesk operators enter problem tickets directly into the database. Spelling, grammar and punctuation are inconsistent. The style is terse and the vocabulary very specialized.

Given the inconsistency of problem ticket sentence structure, we determined that natural language processing techniques were out of the question. Nevertheless, significant information might still be obtained from a tabulation of word occurrences in each problem ticket. After eliminating common stop words and (high- and low frequency) non-content-bearing words, we represented the text data set as a vector space model; that is, we represented each problem ticket as a vector of certain weighted frequencies of the remaining words [7]. We used the **txn** weighting scheme [6]. This scheme emphasizes words with high frequency in a document, and normalizes each document vector to have unit Euclidean norm. For example, if a document were the sentence, "We have no bananas, we have no bananas today," and the dictionary consisted of only two terms, "bananas" and "today", then the unnormalized document vector would be [2 1] (to indicate two bananas and one today), and the normalized version would be: $\left[2/\sqrt{5}, 1/\sqrt{5}\right]$.

As our primary tool for automated classification, we used the k-means algorithm [2], [4] using a cosine similarity metric [5] to automatically partition the problem tickets into k disjoint clusters. The algorithm is very fast and easy-to-implement. See [5] for a detailed discussion of various other text clustering algorithms. The k-means algorithm produces a set of disjoint clusters and a *centroid* for each cluster that represents the cluster mean.

Once a text categorization has been determined, we employ a novel search strategy to find FAQ's within each cluster. We first ask the user to tell us the appropriate level of detail, which translates to the number of identical keywords the examples comprising an FAQ must have (M). We then employ a simple search strategy to find all the examples within each cluster that share at least M keywords. After removing those FAQ's that containing too many overlapping examples with larger FAQ's we present to the user those FAQ's which occur most frequently.

In Section 2, we describe our text classification methodology in some detail. In Section 3, we describe our tunable search algorithm for finding groups of problem tickets within each cluster that have sufficient similarity to be an FAQ. In Section 4, we present an initial case study of how we used our software to discover FAQ's for an IBM customer and further describe how our approach has been incorporated as a key component in the IBM Virtual Helpdesk. Finally, in Section 5, we discuss conclusions and possible future enhancements.

2. TEXT CLASSIFICATION METHODOLOGY

In this section we describe the general process used for creating a classification of the problem ticket information. This classification is a critical step in the process of finding knowledge gaps between the problems and solutions. In the absence of such a classification we would need to do a detailed analysis of individual problem tickets against individual solution documents. Typically this is accomplished by sampling the problem tickets and manually categorizing and counting. Depending on the size of our problem ticket data set, this could be a daunting, expensive and error prone undertaking. This approach usually leads to the most common questions being addressed, but can miss many important problems and sometimes whole categories. Particularly, as more emphasis is placed on leveraging knowledge bases to augment service and support operations more and increasingly complex issues must be addressed within the knowledge base. The approach to manually categorize and count simply does not scale to this problem.

We use the k-means text clustering algorithm to obtain an initial clustering of the data. First, a dictionary of terms is generated by finding those words in the text that occur with the highest frequency. A generic *stop word* list is then subtracted to

eliminate common non-informative words such as "and" and "the". What usually remains, in addition to the informative words, is a set of proper names. Some of these, such as "Lotus", may be useful features for clustering. Others, such as "Bob", may be a distraction to the clustering algorithm. Our software provides features for automatically locating proper names (via capitalization) and for manually refining the dictionary to improve clustering performance. A stemming algorithm [3] is employed to combine all variants of a word into a single term (e.g. print, prints, printer, printed, etc.). Two word phrases are discovered by looking for pairs of words that occur together in sequence or with only stopwords in between. Both words and phrases are treated equally and the only the most frequently occurring terms (words or phrases) are retained in the dictionary. Each problem ticket is converted into a vector of floating point values by counting in each problem ticket the number of occurrences of each dictionary term. This integer vector is then normalized to have unit Euclidean norm. The k-means algorithm is then executed starting with k random seeds selected from the population of problem tickets. The distance metric employed is the cosine similarity metric. Thus two points are considered identical if the cosine of the angle between them is 1.0 and considered most dissimilar if the cosine of this angle is 0.0.

The k-means approach starts by finding the nearest of the k problem ticket seeds to each problem ticket example (using the cosine distance metric). Each problem ticket is then said to "belong" to the cluster corresponding to that nearest seed. For each cluster, a centroid is then calculated, which is the mean of the examples contained in that cluster. In the next iteration, each problem ticket is compared to every centroid and each problem ticket is moved to the cluster of the nearest centroid. The centroids are then recalculated and the process continues until an iteration does not result in any cluster membership changes from the previous iteration. The result of the clustering is the set of centroids (one per cluster) along with an integer vector describing which cluster each problem ticket example belongs to.

After clustering is complete, a final "merging" step takes place. In this step, two or more clusters which are each dominated by the same keyword (dominated means that 90% of the examples contain this keyword) are merged into a single cluster and a new centroid is calculated based on the combined example set. The reason we do this is to avoid arbitrarily separating similar examples into separate subsets before we have a chance to count the Frequently Asked Questions in each class.

Finally each cluster is given a name that describes it. In general of a single term dominates a cluster this term is given as the cluster name. If more than one term dominates then the most frequent term in the cluster is given as the cluster name. If no term dominates then the most frequent term in the cluster becomes the first word in the name and the remaining set of examples (those not containing the most frequent term) are analyzed to find the dominant term. If a dominant term is found then the name is complete, otherwise the process continues for up to four word length names. Beyond four words we simply call the class "Miscellaneous".

3. DISCOVERING FAQ'S

Text clusters generated by k-means have not proven to always be sufficiently homogenous enough to serve as FAQ's. Therefore an additional search step is required to extract subgroups within each cluster that have the requisite homogeneity of word content. The goal of our search is to find subsets of examples within each cluster that all share a requisite number of terms. Our approach to finding this subset is a greedy search over a limited subset of the dictionary. Below is the outline of the search algorithm.

Table 1. FAQ Generation Algorithm

1. Identify a dictionary, D, of frequently used words in a text data set T.
2. Count the occurrences of dictionary words in documents of data set T.
3. User selects level of detail, M, depth of search, S, confidence, C, and overlap, P.
4. For each cluster:
4.1. Sort the dictionary terms in order of decreasing occurrence frequency within the cluster.
4.2. Select the top, S dictionary terms. This is the search space.
4.3. Choose a set of L terms from the search space. For each possible choice of L terms from S:
4.3.1. Find the number of examples containing all L terms.
4.3.2. If the number of examples containing all L terms is greater than C and if the overlap between this set and all other sets is less than P, then this set of examples becomes an FAQ.
4.3.3. Give a name to each FAQ based on the relevant terms in the order in which they occur most often in the text.
5. List each FAQ found along with its associated cluster and size sorted in order of decreasing size.

In the rest of this section we discuss each step of the above algorithm in more detail. Step 1 and Step 2 were accomplished as part of the text clustering described previously. In step 3 we allow the user to adjust the parameters of the FAQ search:

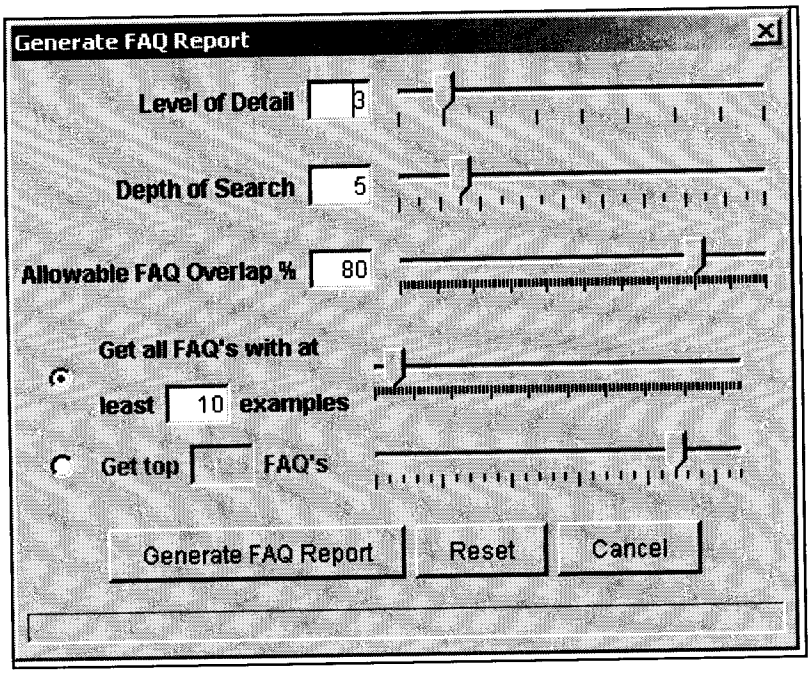

Figure 1. User Input Form for FAQ Search

The level of detail indicates the number of keywords (M) that need to be in common for the subset to qualify as an FAQ. The depth of search (S) indicates how many keywords in each cluster are considered. Only the most frequently keywords of each cluster will be counted in finding potential FAQ's. This helps keep the search tractable for large data sets while still finding all of the large FAQ sets. The allowable overlap (P) indicates how many examples two different FAQ's are allowed to have in common, since our search strategy allows problem example to fall within multiple FAQ's. Finally the user indicates whether they want to generate a specific number of FAQ's (C=1 initially and then is adjusted after all FAQ's are found and sorted by size), or get all FAQ's having a minimum number (C) of supporting examples. Once these inputs are provided it is a simple counting exercise to determine the top FAQ's for a given data set and clustering. An example of such an FAQ set is shown below.

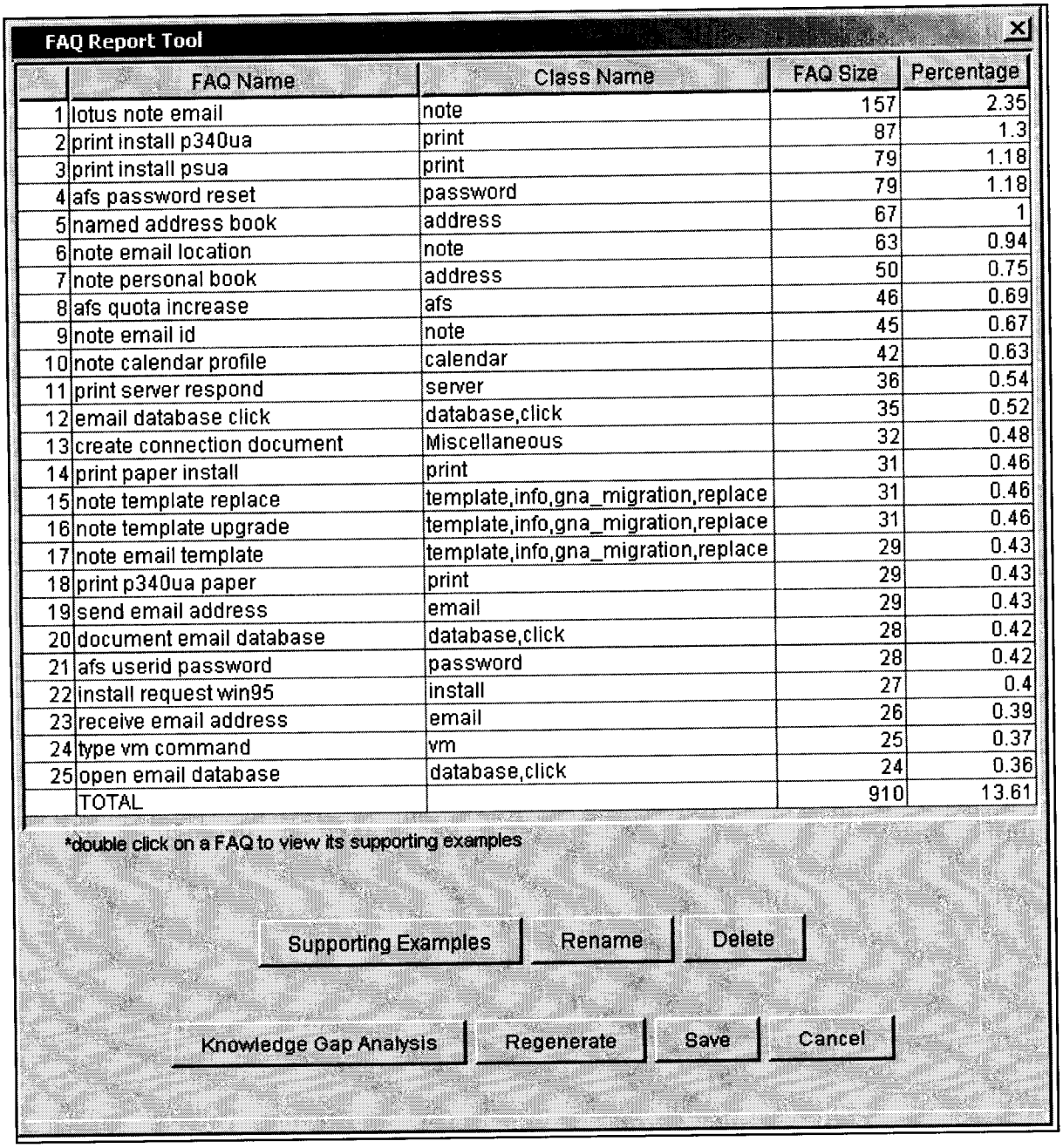

Figure 2. FAQ Generation Output

Each FAQ is given a system-generated name based on the keywords that define it. The keywords are listed in the order in which they most often appear in the document text, in order to create a natural sounding phrase. Along with the FAQ name, the corresponding cluster (class) of each FAQ is given. Once FAQ's have been generated the user is given the opportunity to view the supporting examples, rename any FAQ, or perform a Knowledge Gap Analysis against a knowledge base solution set [8]. The user may also regenerate the FAQ's using different inputs to achieve more specific/general problem groupings, or to do a deeper search, or one with more/less overlap between FAQ's.

4. CASE STUDY AND APPLICATION

In our first Case Study, an IBM customer provided 36,000 problem tickets from their helpdesk. In the first phase, the problem tickets were clustered initially into 30 categories. Several of these categories were then automatically merged based on dominating keyword similarity into a set of high level clusters shown below, sorted in decreasing order of size (number of examples).

	Class Name	Size
7	lotus_note	8267 (22.91%)
12	window	6337 (17.56%)
13	trax	4959 (13.74%)
8	securid	4312 (11.95%)
9	lan	2994 (8.30%)
10	microsoft_office	2868 (7.95%)
5	laptop	1594 (4.42%)
1	aws	1495 (4.14%)
3	internet	1226 (3.40%)
11	password	963 (2.67%)
4	list	547 (1.52%)
6	drive	299 (0.83%)
2	modem	222 (0.62%)
	TOTAL/AVERAGE	36083

Figure 3. Clustering Results

Each class was given a name based on the most frequently occurring keyword among the examples in the class. Next, FAQ generation was employed, using the default inputs (M = 3, depth of search = 5, allowable overlap = 80%, and at least 10 examples in each FAQ). The first 25 FAQ's are shown below.

FAQ Report [Top]

1	lotus note email	lotus_note	5668	15.71
2	lotus subject list	lotus_note	3486	9.66
3	trax personal submitting	trax	2377	6.59
4	window dial remote	window	2025	5.61
5	securid card verify	securid	1575	4.36
6	securid verify cis	securid	1433	3.97
7	securid card cis	securid	1419	3.93
8	microsoft office word	microsoft_office	1376	3.81
9	lan server novell	lan	1359	3.77
10	securid passcode card	securid	1175	3.26
11	securid passcode verify	securid	1168	3.24
12	window print network	window	1165	3.23
13	internet explorer proxy	internet	812	2.25
14	securid passcode cis	securid	788	2.18
15	laptop dell cpi	laptop	533	1.48
16	trax issue submit	trax	415	1.15
17	window password reset	password	372	1.03
18	window subject list	list	348	0.96
19	laptop dell ltss	laptop	325	0.9
20	window password account	password	191	0.53
21	laptop battery dell	laptop	172	0.48
22	aws engagement atl	aws	167	0.46
23	microsoft excel document	microsoft_office	159	0.44
24	aws atl left	aws	136	0.38
25	laptop pcmcia modem	modem	132	0.37

Figure 4. FAQ Results

Using this approach it took only a few hours to create an FAQ list that had taken three days to create manually through reading problem tickets. The quality of the automatically generated list was considered by the customer to be equal or better than that generated manually. The success of this exercise convinced the IBM HelpDesk facility in Tampa to use the FAQ tool on a regular basis on all of its customer accounts. After several months of testing and assuring that the technology was consistent with the top problems being reported, it was consistently reported that the tool was about 100% accurate in determining the top problems and FAQ's from the 35,000 to 40,000 calls the Tampa helpdesk received each month. After further testing with other customers being supported out of Tampa as well as Canada confirmed these results, the IBM Virtual Helpdesk team decided to incorporate this approach into their process for creating customized helpdesk solutions for customers.

For IBM Virtual HelpDesk, an IBM group that creates and customizes eSupport tools for companies, the benefit of using automated FAQ generation has been primarily in the ability to keep solutions current with existing incoming calls. IBM Virtual HelpDesk links to the current help desks that provide support for customers. Experience has shown that the subject of daily customer problems changes over time. Our FAQ generation tool provides the help desks with the ability to run an analysis daily, weekly or monthly in order to keep the websites current without having to spend days or weeks analyzing the information, by which point the analysis might already be out of date.

Virtual HelpDesk takes the top problems by category (such as Lotus Notes, MS Word etc.), that the tool has created and uses this categorization to customize the customer's websites.

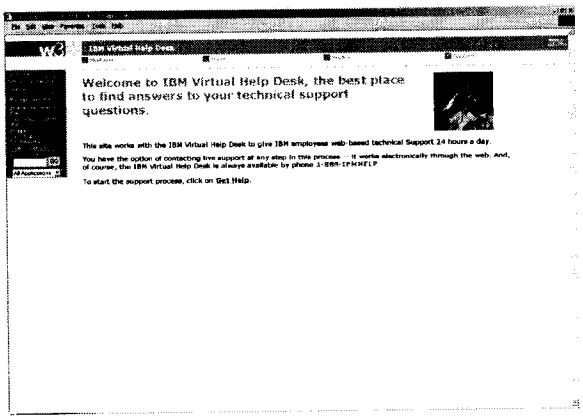

Figure 5. IBM Virtual Helpdesk Main Screen

Based on the categories (i.e. clusters) that the tool found the next page of Virtual HelpDesk is created

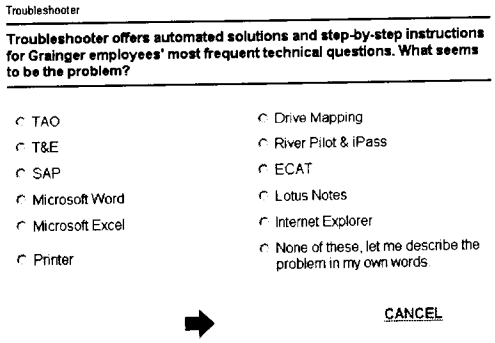

Figure 6. High Level Categories in VHD

The FAQ's generated for each category provide the basis for the next screen. Note that the FAQ descriptions have been expanded by the analyst based on the text descriptions found in the most representative examples of each FAQ.

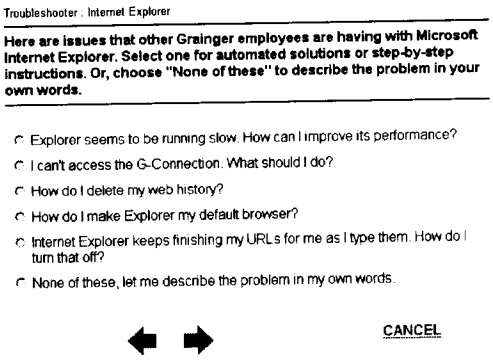

Figure 7. FAQ's in VHD

The end user then selects the appropriate solution, or if their solution is not listed they can search several solution databases or open a help request with the help desk via the web.

The ongoing FAQ maintenance and update process can now be done in most cases in a matter of minutes, where before such an analysis would have taken days.

5. CONCLUSIONS

We have described a novel approach for automatic FAQ generation from helpdesk problem tickets. We have also shown how we use this approach in practice to greatly reduce the time required to find FAQ's vs. manual methods. The great advantage of our approach over other techniques is that it allows the user to customize the search to achieve the right level of FAQ specificity for a given problem domain.

Possible future enhancements to this algorithm would include the ability to specify different levels of FAQ specificity for different problem clusters. This would help to alleviate a problem we have encountered on some data sets where some FAQ's from particular clusters are too general when compared to FAQ's from different clusters.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the work of Michael Sanchez, a summer employee at IBM Almaden, who wrote much of the software for FAQ analysis.

REFERENCES

[1] Brachman, R. and Anand T. (1996). The Process of Knowledge Discovery in Databases. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining,* Chapter 2, pages 37-58. AAAI/MIT press.

[2] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* Wiley.

[3] Frakes, W. (1992). Stemming algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms,* pages 131-160. Prentice Hall, Englewood Cliffs, New Jersey.

[4] Hartigan, J. A. (1975) *Clustering Algorithms.* Wiley.

[5] Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms,* pages 419-442. Prentice Hall, Englewood Cliffs, New Jersey.

[6] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management,* 4(5):512:523.

[7] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Retrieval.* McGraw-Hill Book Company.

[8] Spangler, S. and Kreulen, J. (2001). Knowledge Base Maintenance Using Knowledge Gap Analysis. Proceedings of the Seventh ACM SIGKDD-2001. Pages 462-466. ACM, New York, NY, 2001.