eResponder: Electronic Question Responder

David Carmel and Menachem Shtalhaim and Aya Soffer
IBM Haifa Research Laboratory
MATAM, Haifa 31905, Israel
{carmel,ayas,menachem}@il.ibm.com

Abstract

With the complexity of systems increasing, support centers are flooded with questions submitted by e-mail, by Web, or by phone. This paper describes eResponder, a system which provides an integrated solution for automatic responses to user questions. eResponder stores question and answer pairs that have previously been asked. These pairs can be used to either provide an immediate response to user questions, or to assist customer service representatives in drafting new responses to similar questions or to yet unanswered questions. Users submit free text questions to the system via one unified interface. When a new question arrives, the system searches its databases for similar questions as well as for relevant answers to this question and finds the most relevant Q&A pair based on these measures. eResponder provides a relevance feedback mechanism and an answer summarizer to assist CSR’s in creating new responses. The results of an experiment conducted to evaluate the system performance show that the combination of independent question and answer scores yields high precision search results.

1 Introduction

With the complexity of systems increasing, and with user expectation for immediate assistance, support centers are flooded with a large number of questions submitted by e-mail, via their Web site, or by phone. Companies that attempt to answer all of these questions need to allocate substantial resources, in terms of customer service representatives (CSR). NationsBank, for example, learned just how daunting large quantities of email can be. In 1996, as it’s electronic banking clientele grew from 50,000 to 250,00, the number of messages soared from a few hundred to 20,000 a month. The following year, the bank had to hire about 100 people to cope with the external load [4].

A large number of questions submitted to call centers, turn out to be repetitive and address common problems that can easily be resolved and have been answered several times before. In order to reduce the effort required to answer new questions, a company should be able to capitalize on previous work of its CSRs, which have already answered many similar questions before. The accumulated knowledge encapsulated in this collection of previous answers could thus be exploited for better customer support.
A simple solution for this problem, used by many companies (e.g., Microsoft [3], Intel [2]), is to provide a Web interface to a collection of frequently asked questions and answers (FAQ). This collection can be searched and browsed by customers to locate answers to frequent problems which have been faced by many customers before, without any manual intervention of an expert, thus, reducing significantly the overload on their call center.

In addition to company FAQ services, there is also a large number of Web sites which are used as forums for asking experts questions or for asking fellow users questions [5, 1, 8]. These collections of questions and answers are usually stored in knowledge bases. A user with an unresolved problem may search these collections for previously asked questions of a similar nature. If she does not find a satisfactory answer, then she can post a new question to the community and a fellow community member will hopefully post a response. These knowledge bases and Web sites are very large cooperative systems in which a geographically distributed community interested in common domains can assist others in the community.

There are some drawbacks to these solutions. Users need to submit a query that is almost identical to the question stored in the FAQ or forum in order to receive the proper answer. A search for a slightly different query, which describes the user’s problem in different terms, will probably fail. Moreover, if the user’s problem has not been handled before or is not stored in the knowledge base, most of these systems will retrieve non-relevant answers, leaving an unsatisfied customer. Furthermore, if you do not find an answer in the FAQ, then you will most likely need to resubmit your question to a human CSR. From a user-centered point of view, it is desirable to deal with only one entry point. Users should be able to submit their questions and either receive an immediate response if one exists, or get an answer from a CSR by email later.

eResponder is a system that provides an integrated solution for automatic responses to user questions. The underlying idea is to store in a Q&A knowledge base question and answer pairs that have previously been asked which can be used to either provide an immediate response to users questions, or to assist CSRs in drafting new responses to yet unanswered questions. Users submit questions to the system via one unified interface. When a new question arrives, the system searches its databases for similar questions as well as for relevant answers to this question. From these questions and answers, eResponder will follow one of the following paths:

- Identify similar questions in the knowledge base with their associated answers, which can automatically be returned to the user.
- Infer enough information from a set of similar questions and relevant answers to generate an answer that can automatically be returned to the user.
- Create a draft answer that can be manually corrected by a CSR and emailed to the user.
- Recognize that it has no relevant information and pass the question to a qualified CSR that can answer this question and also insert it into the Q&A knowledge base for future use.
eResponder is based on advanced information retrieval (IR) techniques for computing similarity of new questions to previously asked ones based on their full text description as well as for computing the relevance of previous answers to this new question. The goal is to avoid previous questions whose answers are not relevant or perhaps locate previous relevant answers although the original question is not similar to the new question.

The user’s query is specified using free text. Query terms are stemmed and stop words are removed. The Q&A pairs with the most similar questions and most relevant answers are retrieved from the Q&A knowledge base. Each pair is assigned a confidence level, which reflects its relevance to the new question.

eResponder can function in two modes. In the online mode, the queries are submitted via a Web page. If similar Q&A pairs are located, then the results are returned as an HTML page. If none of the results satisfy the user, then she can resubmit the query to a CSR that will either generate a response based on previous questions or write a new response and send it to the user by email. In the offline mode, the question is submitted via email. If there is an almost identical question, then a response will be generated automatically and sent to the user, otherwise it will be added to the CSR queue as in the online mode.

The CSR can employ a relevance feedback mechanism for query refinement and resubmit the query to the system. In addition, he can ask eResponder to create a draft response based on a few selected answers from a set of relevant pairs suggested by the system. The draft generator uses summarization techniques to identify the sentences, which are most similar to the new question, as well as the most significant sentences in the set of selected answers. The set of significant sentences can be used by the CSR to synthesize a reply to the user’s question.

The rest of the paper is organized as follows. Section 2 presents a sample session using eResponder. Section 3 describes the eResponder architecture. Section 4 provides some experimental results. Section 5 concludes the paper.

2 eResponder Sample Session

In this section we present a sample session using eResponder in online mode with a database of 90 Q&A pairs concerning the Mapuccino software developed at IBM research lab in Haifa [7]. Figure 1 is a screen shot of the user interface for choosing a knowledge base and submitting a query. The user is asking how to print a map (created by Mapuccino).

In this case there are perfect matches to this query, and thus the system will return a list of solutions to consider as seen in Figure 2. The answers can be viewed by clicking on the relevant questions. If none of the answers are satisfactory, the user has an option to pass the query to a CSR by pressing the “Resubmit” button.

Assuming, that the user is unsatisfied, or that there were no perfect responses, the token would now be passed to a CSR. Figure 3 is the CSR view of the query results. The 10 top responses are presented with their associated question similarity (Q-similarity), answer relevance (A-relevance), and combined score (Total Score). The CSR could mark
some of the questions as relevant or non-relevant using the left pane and resubmit the query by pressing the “Refine” button. In this example, this is not necessary, as there are several good answers. There are in fact 3 answers with an A-relevance of 100, and the CSR selects these three as the basis of the draft.

Note that the question in the third response chosen by the CSR is not very similar to the user’s question (Q-similarity of 57.5), however the answer is judged to be very relevant (A-score 100). The CSR has thus selected it to be included in the new response.

Pressing the “Generate Draft” button, opens Netscape composer as seen in Figure 4. The subject line contains the text of the user’s new question. For each selected Q&A pair, first the overall score (confidence level) is displayed followed by the text of the original question and the answer. The terms that are common to the new question and the old question/answer are in bold face. Each sentence in the Q&A pair is either light gray, dark gray, or black according to its significance to the new question and to the answer. Light gray indicates that the sentence is not relevant, dark gray indicates that the sentence is somewhat relevant, and black indicates that the sentence is very relevant. The CSR can synthesize a new response by selecting the most significant sentences from the set of answers and tailoring it for the new question.

Note that in this case all of the sentences in black in fact discuss printing the Mapuccino map. In particular notice the answer to question 3. Even though the original question did not specifically ask about printing the map, the answer does include the proper instructions. Since eResponder evaluates the questions and answers separately it was able to find this answer and indicate this to the CSR for his consideration.
3 eResponder Architecture

eResponder is a Java-based system composed of the following components:

1. A validation tool that dynamically indexes question-answer pairs and incrementally adds them to one or more Q&A knowledge bases.

2. A highly precise search engine that compares newly received questions to existing ones (as stored in the system knowledge bases), and returns a list of most related Q&A pairs ranked by degree of confidence.

3. A relevance feedback mechanism, which enables the CSR to provide feedback to the system by marking relevant and non-relevant Q&A responses. eResponder applies query refinement to transform the original query into a new query that reflects the CSR feedback [11, 6]. The refined question is then resubmitted to the system.

4. A response draft generator, which assists in synthesizing a new answer from a given set of old relevant answers. The generator summarizes several answers marked as
relevant by the CSR by identifying the most significant sentences in each answer. These sentences can be used to draft a new answer to the new question.

Figure 5 describes the components of system architecture. In the following these basic components are described in more detail.

3.1 Storage and Indexing

Storage and indexing is the process that receives a Q&A pair and stores it for future retrieval in one or more Q&A knowledge bases. Each knowledge base contains a Q&A database, called Q&A dB, and two inverted indexes, one contains all of the question terms, while the other contains all of the answer terms. Each Q&A pair is associated with a unique identifier that is used as its key inside the Q&A dB. The unique identifier associated with the Q&A pair is used as the “filename” in the inverted indexes to identify the text of the associated question/answer. The Q&A database holds the Q&A pairs along with any additional information associated with these pairs. This information might include a list of categories when the pair was first categorized by an outside categorizer, a URL address if the answer is stored as a URL page, the date of submission, the email address of the submitter, etc.
eResponder uses Juru, a Java search engine developed at IBM research lab in Haifa, for indexing and searching the knowledge bases. Juru extracts for each question or answer a canonical representation, called a profile, which consists of a vector of indexing units of representative terms. The terms are stemmed and stop words are removed. Juru uses Porter’s stemmer [10] to lemmatize the index terms. Each knowledge base has its own stop word list of very frequent words to be filtered out from the text before storing it. Terms used by Juru for indexing are lemmatized words and lexical affinities that we identify as closely related terms frequently found in proximity to each other. It has been described elsewhere [9] how lexical affinities, when used as indexing units, improve precision of search results as compared to single words, especially in the context of IR systems.
3.2 Retrieval

The retrieval process treats new incoming questions as “free text queries” and retrieves a ranked list of the most relevant Q&A pairs from a set of predefined knowledge bases. Juru’s inverted indexes and its retrieval mechanism are used to find the questions that are most similar to the new question as well as the answers that are most relevant to this question.

Unfortunately, there is no precise distinction between relevance and similarity in IR systems. There are many cases when a relevant answer is not necessarily similar to the query. While similarity among questions can be easily measured, it is not clear how to evaluate the relevance of answers to the query. In the traditional vector-space model, the relevance of a document to a given query is determined according to their vector-space similarity. Even in the probabilistic model, which tries to measure the relevance probability of a document to a given query, the probability approximation is actually based on similarity between the document and the query. In the current version, eResponder does not distinguish between relevance and similarity measures. There is need for a better understanding of the difference between these two characteristics. This study is left for further research.

eResponder first creates a profile for the new question, using the same process used in the indexing process, and computes a textual similarity score for each question and each answer in the knowledge base with respect to this profile. Each term in the profile is associated with a weight \( w_i \) computed by a \( tf-idf \) formula as follows:

\[
    w_i = tf_i \cdot \log\left(\frac{1}{pr_i}\right)
\]
where \( f_i \) is the frequency of term \( i \) in the profile, and \( pr_i \) is the probability of term \( i \) in the entire knowledge base.

The confidence of any question and answer is determined by measuring the cosine of the angle between the profile of the new question and the corresponding profile of the question or answer respectively. The overall confidence value for the Q&A pair is a weighted linear combination of these two scores.

The result of the retrieval procedure is a ranked list of Q&A pairs. This ranking paradigm is only useful for comparison purposes. However, since eResponder must evaluate each pair independently to decide whether it can be automatically returned to the user as a proper answer, ranking does not fit this requirement. Therefore, every Q&A pair is independently scored and judged according to its similarity to the user’s question. Juri’s score is normalized by eResponder to a number between zero and 100. The zero value is assigned to absolutely non-relevant text (a question or an answer with no common terms with the query), while a score of 100 is assigned to a text that is identical to the query.

3.3 Relevance feedback

The user’s query may often not be specific enough to enable retrieval of highly relevant answers. This is especially the case when the question is submitted via a web-based form where users tend to type very little. In addition, natural language ambiguity often causes people to describe concepts in their questions in a different manner than others have described the same concepts in their questions or answers. Therefore, in some cases, the Q&A pairs, which were identified by the system as most relevant, may not be judged as such by the CSR.

The low precision problem has long been recognized as a major difficulty in information retrieval systems. To handle it, most search engines apply some form of query refinement, which transforms the original query into a new query that better reflects the user needs. The query is expanded using words or phrases taken from a set of documents marked as relevant for the given query. Specifically, given a set of relevant documents and a set of non-relevant documents, an “optimal query” is derived by addition of terms characterizing the relevant documents and subtraction of terms characterizing the non-relevant documents. In addition, the weights of the query terms are modified to reflect their relative importance in the matching between the refined query and the marked documents.

In eResponder the CSR has the option to refine the original question iteratively by marking the retrieved Q&A pairs as relevant or non-relevant. The original question is then modified as explained above and resubmitted to the system.

3.4 Automatic Draft Generator

If there are retrieved Q&A pairs with a score which is a higher than a system-set threshold, then these results are returned to the user. For questions of a very repetitive nature, the
user should find a suitable response in this set.

If there are no Q&A pairs with high enough scores, or if the user did not find a suitable response, then eResponder constructs a draft response which is to be edited by the CSR. The CSR first marks the most significant Q&A pairs for the new question. Next, eResponder estimates the significance of each sentence in these answers. Each sentence is assigned two scores. This first score indicates the relevance of this sentence to the problem (a similarity score between the sentence and the query). The second score indicates the relative significance of this sentence in the set of sentences in the marked answers. Using these scores eResponder indicates to the CSR which sentences are recommended for inclusion/exclusion in the response to the new question using different gray levels (see Figure 4). eResponder also highlights the original question terms in the retrieved questions and answers. This also assists the CSR in selecting the sentences to include in the new answer.

We compute the two sentence scores by a modification of the methods described in [12].

1. Similarity to the new question. The similarity between sentence $s$ and query $q$ is measured using a vector space model similarity function.

$$
SQ(s, q) = \frac{\sum_{i=1}^{M} w_{i,s} w_{i,q}}{M}
$$

where $w_{i,j} = t_{f_{i,j}} \cdot \log \left( \frac{N}{N_{f_i,c}} \right)$, $t_{f_{i,j}}$ is the term frequency of term $i$ in sentence $j$, $t_{f_i,C}$ is the term frequency of term $i$ in the corpus $C$, $N$ is the number of terms in the corpus, and $M$ is the number of terms in sentence $s$.

2. The internal weight of the sentence. The idea here is that sentences that contain terms that appear frequently within the entire set of selected answers are considered the most significant. The score of sentence $s$ in answer $a$ in answer set $A$ is computed as follows:

$$
SI(s, a, A) = \frac{\sum_{i=1}^{M} w_{i}}{M}
$$

where $w_{i} = t_{f_{i,s}} \cdot t_{f_{i,a}} \cdot \bar{t_{f_{i,A}}} \cdot \log \left( \frac{N}{N_{f_i,c}} \right)$

Where $t_{f_{i,s}}$ is the term frequency of term $i$ in sentence $s$, $t_{f_{i,a}}$ is the term frequency of term $i$ in answer $a$, $t_{f_{i,A}}$ is the average term frequency of $i$ in the set of selected answers $A$, $t_{f_{i,C}}$ is the term frequency of $i$ in the corpus $C$, $N$ is the number of terms in the corpus, and $M$ is the number of terms in the sentence $s$. If a term appears in only one sentence, then its weight is set to zero. The weight of lexical affinities is boosted in order to increase their significance in the sentence score.

The vector space similarity measure, tends to be biased towards short sentences. This abnormality has been described in [14]. In order to avoid this bias, the actual normalization for sentence length used by eResponder is based on the pivoting method described in that paper. In particular, rather than dividing by the sentence length $M$, we divide by a linear combination of $M$ and the length of an average sentence in the database. This is required to reduce the effect of abnormally short sentences.
4 System Evaluation

In order to evaluate eResponder, it is necessary to have some benchmark for such systems. One option would be to use the TREC question answering benchmark [15], which provides a set of fixed questions and the set of documents which best answer these questions. In this model, however, the questions are not part of the corpus of documents, and thus it is not suitable for a system that stores both questions and answers such as eResponder.

We are not aware of any existing benchmark for Q&A pairs. We thus used a set of Q&A pairs that are available on the NSF Ask a Scientist or Engineer Web site [5]. This is a collection of questions which youngsters have asked scientists along with their responses. The questions cover many areas of science including biology, chemistry, earth science and more. The Web site provides a search engine to search the Q&A database.

We downloaded the Q&A pairs from the NSF site and inserted them into the eResponder system. We compared the results of searching for the same information using the NSF search engine and eResponder. We submitted 30 queries to both systems and counted the number of relevant answers returned by each. Table 1 shows some examples of the questions submitted.

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>What happens to your body when you dream?</td>
</tr>
<tr>
<td>How do scientists clone animals?</td>
</tr>
<tr>
<td>How does the whale use its blubber?</td>
</tr>
<tr>
<td>How many trees are chopped down a year to make paper?</td>
</tr>
</tbody>
</table>

Table 1: Some of the questions submitted to eResponder and NSF Ask a Scientist.

We compared the number of relevant Q&A pairs which appeared in the first 3 responses (top@3), the first 5 responses (top@5), and the first 10 responses (top@10). Figure 6 shows the comparison of these parameters computed for the two systems.

The results show that eResponder is much more precise than the search engine used by the NSF site. When the query was almost identical to a question stored in the database, both systems had no problem locating the relevant Q&A pair. However, for free text queries, eResponder functioned much better. This can be attributed to several reasons. First, eResponder uses IR techniques for search which include term weights, lexical affinities, and an advanced ranking algorithm. On the other hand, the NSF site's search engine seems to be based on Boolean search techniques. Although the questions where submitted in free text form, we tried our best to translate them to comparable Boolean queries. The second reason for eResponder's success is the fact that we index and search the questions and answers independently of each other while the NSF engine seems to store the text of the Q&A pair as one indexing unit. Since the questions are usually much shorter than the answers, the NSF engine seemed to diverge into Q&A pairs that happened to contain the search terms in the text of the answer, although this was not the real topic of the question.

eResponder's method of computing the question similarity and the answer relevance
Figure 6: The precision of eResponder and Ask a Scientist averaged over the same 30 questions

as two separate scores and then combining them, contributes significantly to its high precision. The following example highlights this feature. For the question “what is photosynthesis?” eResponder returns the results shown in Table 2.

<table>
<thead>
<tr>
<th>Similar Questions</th>
<th>Q-similarity</th>
<th>A-relevance</th>
<th>Total Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why are the leaves green?</td>
<td>0</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>Why is chlorophyll green?</td>
<td>0</td>
<td>91.2</td>
<td>45.6</td>
</tr>
<tr>
<td>What do trees use that we breathe out?</td>
<td>0</td>
<td>91.1</td>
<td>45.6</td>
</tr>
</tbody>
</table>

Table 2: eResponder results for the question: “what is photosynthesis?”

Even though none of the questions in the database have common terms with the new question, the system found three relevant Q&A pairs by looking at the answers. The total score is a linear combination of the Q-similarity and the A-relevance. The system administrator can set the weights based on the nature of the Q&A knowledge base. In FAQ files, the question is usually a manually crafted title for the answer, and should thus receive a high weight. On the other hand, in collections of “real” questions and answers, the answer is usually as important as the question and the two should be assigned equal weights.

5 Concluding Remarks

eResponder, a system which provides an integrated solution for automatic responses to user questions, has been presented. eResponder stores in a Q&A knowledge base question and answer pairs that have previously been asked. These pairs can be used to either provide an immediate response to user questions, or to assist customer service representa-
tives in drafting new responses to similar questions or to yet unanswered questions. Users submit free text questions to the system via one unified interface. When a new question arrives, the system searches its databases for similar questions as well as for relevant answers to this question and finds the most relevant Q&A pair based on both these measures. eResponder can thus avoid previous questions whose answers are not relevant to the new question or perhaps locate previous answers relevant to this new question although the original question is not similar to the new question.

eResponder can function in two modes. In the online mode, the queries are submitted via a Web page. If similar Q&A pairs are located, then the results are returned as an HTML page. If none of the results satisfy the user, then they can resubmit the query to a CSR and they will either generate a response based on previous questions or write a new response and send it to the user by email. In the offline mode, the question is submitted via email. If there is an almost identical question, then a response will be generated automatically and sent to the user, otherwise it will be added to the CSR queue as in the online mode. eResponder also provides a relevance feedback mechanism, and an answer summarizer to assist CSR’s in creating new responses.

In the current version, eResponder does not distinguish between question similarity and answer relevance. The same measure is used to compute both, however they are each computed separately. We have already seen that even this distinction is beneficial and improves precision. Using a different measure to compute relevance may improve precision even further. This study is left for further research.

eResponder can use several knowledge bases of Q&A pairs. In order to work with eResponder effectively in this case, a new question must first be assigned to one or more domains using a categorization tool. eResponder will then use one or more domain specific knowledge bases in order to craft the response. The question categorization tool we intend to use is another IBM product [13] and is outside the scope of this paper.

Acknowledgments

We thank Pnina Vortman for initiating this line of research.

References


