A Context-Sensitive Search Mechanism

Omar Hasan[1], Michael E. Atwood[2], Jim Waters[2], and Bruce W. Char[1]
Department of Computer Science[1] / College of Information Science and Technology[2]
Drexel University
Philadelphia, PA 19104, USA
oh23@drexel.edu, atwood@acm.org, jw65@drexel.edu, charbw@drexel.edu

Abstract

Collaborative help systems are a widely used type of information systems. Users seeking information, post questions on a collaborative help system, which are answered by other users who possess the requested information. Collaborative help systems often contain domains of information that are significantly disparate. There may exist several questions in a collaborative help system, which have similar wording but dissimilar context and therefore each one of them has completely different answers. When searching the collaborative help system for a certain question, it is quite possible that a regular search mechanism would return some question that is not in the context that the user wanted. A search mechanism that is sensitive to context, however, would produce the correct result. In this paper we present our context-sensitive search mechanism. The mechanism uses the recent activity of a user as the context of their questions and searches. The mechanism has been implemented in our collaborative help system called Knowledge Exchange.

1. Introduction

Collaborative help systems are a common type of information systems. We use the term collaborative help systems to denote those information systems that have the following characteristics:

1) The information is primarily in the form of questions and answers. Users are permitted to post new questions, which are answered by human sources. The human sources may be individuals explicitly designated as experts or they may be peer users.
2) The information is stored in a knowledge base that "grows" with time as new questions and answers get posted. In addition to posting new questions, users may also benefit from the information previously collected in the knowledge base.

Collaborative help systems are described by Ackerman [2] as "…those help systems that use people as information sources…". Our description further qualifies this definition. Collaborative help systems are also sometimes referred to as "ask an expert systems", "collaborative learning systems" and "discussion board systems".

Collaborative help systems are widely used. Examples of collaborative help systems include Google Answers [8], Experts Exchange [7], Math Forum [13], USENET [10] and Answer Garden [1].

Collaborative help systems usually have a very broad main subject (for example, computer technology) and several sub-topics which are subjects in their own right (for example, game programming, software engineering, wireless networking etc.). This leads to domains of information in the knowledge base that are significantly different from each other. In this kind of knowledge base there may exist many questions, which are similar but have completely different answers because of dissimilar contexts.

Consider the question "What are arrays?". Arrays are a common theme in the subject of computer programming as well as in telecommunications. In each of these subjects, arrays have a very different meaning. In computer programming an array refers to a contiguous collection of data variables whereas in telecommunications an array refers to a series of antennas.

Suppose that an instance of a collaborative help system holds the topics computer programming and telecommunications, and the same or a similar question has been asked under each of the topics. It is quite possible that a user searching for the question "What are arrays?" might find the question that is not in the context that the user wanted.

The user might be able to restrict the search to a specific subject but even one subject may contain

questions that have different contexts. For example the question "Should I use '++i' or 'i++'?" is often asked in C++ programming courses. The correct answer to this question depends on the context in which it is asked. If a student is iterating a 'for' loop, both will have the same effect. However, if a student is using it in an arithmetic expression, it is important to know that 'i++' would not increment the value of 'i' until after the expression has been evaluated.

A context-sensitive search mechanism would clearly be very helpful in this kind of environment. The value of context in information systems has been discussed in several publications ([4], [5], [6], [11], [12], [14]).

Google [9] is a popular search engine with a very successful search mechanism. Google's approach is to rank results in terms of popularity. Results that are more popular receive a higher rank. This approach, however, would not always be best for our problem. If a user is looking for the answer to the question "What are arrays?" in the context of telecommunications, and the same question in the context of computer programming happens to be more popular, the user would not receive the correct result. On the contrary, a context-sensitive search mechanism would produce the correct result.

In this paper we present our context-sensitive search mechanism. The mechanism has been implemented in our collaborative help system called Knowledge Exchange.

2. The Knowledge Exchange collaborative help system

Knowledge Exchange is primarily intended for supporting teaching-learning settings, such as courses taught at colleges and universities. However its use is not limited to teaching-learning settings; it can be used in other environments as well where the roles of those who impart knowledge and those who seek knowledge exist (for example, help desks etc.).

In Knowledge Exchange, questions are posted by students. The questions can be answered by instructors as well as peer students.

The information posted on Knowledge Exchange is stored in a knowledge base and remains available for future reference (until removed by an instructor).

2.1. Organization of information

In Knowledge Exchange information is organized under a tree-structured hierarchy of topics. The topic hierarchy can be created and modified over time by the instructors.

The main subject is represented by the root topic, which is the ancestor of the rest of the information in the tree.

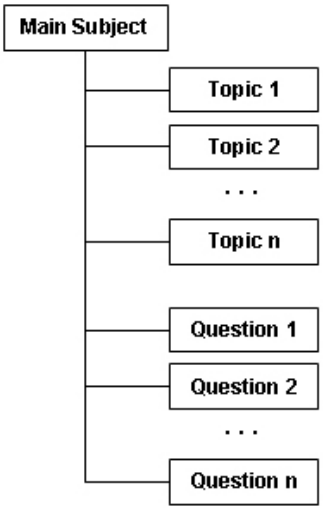

Figure 1. Structure of information under the root topic (the main subject).

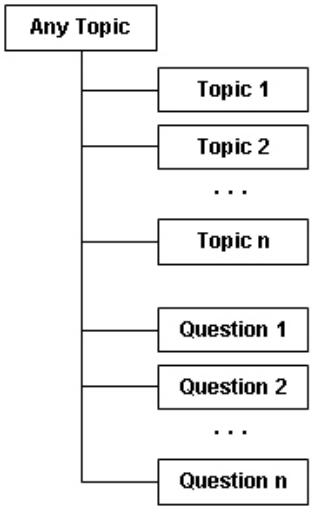

Figure 2. Structure of information under any topic.

Each question ideally exists under the most relevant topic. One way to accomplish this is to encourage the users to post a question under the topic they consider the most relevant to that question. The instructors have the ability to move a question from one topic to another topic when they feel that it is not under the most relevant topic.

A question contains three folders: 'Answers from Experts', 'Answers from Peers' and 'Follow-Up Questions'.

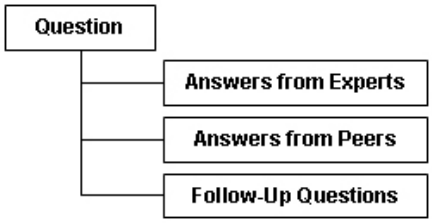

Figure 3. Folders under a question.

The 'Answers from Experts' folder contains any answers to the question that have been posted by the instructors. The 'Answers from Peers' folder contains any answers to the question that have been posted by the students. The 'Follow-Up Questions' folder contains questions that are spawned in response to the answers to the original question. The questions in the 'Follow-Up Questions' folder are themselves full-fledged questions.

2.2. Using Knowledge Exchange

2.2.1. Finding information. Users have the option of using three different methods for finding the information that they need:

1) By browsing the knowledge base.
2) By looking under the specialized folder 'Recently Posted Questions' which contains links to the questions that have been posted in the past seven days (the number of days may be adjusted).
3) By using the context-sensitive search. As with a regular search mechanism, the user only provides keywords. The context is obtained implicitly by the system.

Figure 4. A screen from Knowledge Exchange showing the contents of the root folder.

2.2.2. Asking a new question. If the information that a student needs is not available in the knowledge base, he/she can ask a new question.

If the instructor's answer does not satisfy the student, he/she can also ask a follow-up question to the previously posted question.

2.2.3. Answering a question. Both instructors and students can post answers to a question. The 'Answers from Experts' folder of a question holds the answers given by the instructors. The 'Answers from Peers' folder holds the answers posted by students.

2.3. Architecture and technology used

Knowledge Exchange is a web-based multi-tiered system built with Java, JSP, JavaScript, HTML and SQL. Extreme programming [3] practices were used for its development.

3. The context-sensitive search mechanism

Our context-sensitive search mechanism is based on the premise that the behavior of a user in the recent past often depicts the current behavior of the user. For example in Knowledge Exchange, if a user was recently browsing information on music, it is more

likely that their current search is related to music, rather than some other subject.

The information that a user was looking at in Knowledge Exchange before posting a question or search is taken as its context.

The mechanism comprises of the following key steps:

1) When a user posts a new question, its context is determined and saved in the knowledge base along with it.
2) When a search is submitted, a list is made of all the questions in the knowledge base that contain those keywords. Additionally, the context of the search is also determined.
3) For each of the questions in the list, it is ascertained how much its context is similar to the context of the search.
4) The search results are presented in descending order such that the question, whose context has the highest similarity with the context of the search, is listed first.

The following sections describe the mechanism in detail.

3.1. Maintaining a nodes-visited history

As a user navigates through the knowledge base, a history of the nodes that are visited is maintained (topics and questions are considered as nodes). This history is a record of where the user has looked for information. The *nodes-visited history* is limited to the recently visited nodes. In the current implementation the limit is 20 last nodes. This number can be experimented with in future work.

3.2. Construction of a nodes-visited weights table

When a question is posted, a *nodes-visited weights table* is derived from the nodes-visited history existing at that point. The derived table is captured in the knowledge base along with the question.

Similarly when a search is requested a nodes-visited weights table is also derived from the nodes-visited history. This nodes-visited weights table is submitted to the Knowledge Exchange search-engine along with the keywords.

The nodes-visited weights table is used as the context of a question or search (details will be given in the following section). In this section we discuss the construction of this table.

The table has two columns: node and weight. To derive the table from the current nodes-visited history, each of the nodes in the history as well as all the ancestors of each of those nodes are listed in the node column. There are no recurrences.

A weight is assigned to each node and is listed against it in the weight column of the table. The weight assigned to a node depends on two factors: number of times visited and depth in the knowledge base tree hierarchy.

Number of times visited: Each time a node is visited, all its ancestors are also considered visited. For example if a user follows the path A(ROOT) → B → C → D → C, the *number of times visited* of each node would be as follows: A: 5, B: 4, C: 3, D: 1.

Depth factor: Each depth of the knowledge base tree hierarchy has a pre-assigned constant value called the *depth factor*. The value assigned to each depth is 2.5^{depth}, for example the value assigned to depth 0 (the root) is $2.5^0 = 1$, to depth 1 is $2.5^1 = 2.5$, to depth 2 is $2.5^2 = 6.25$ and so on. The idea is that the further down the tree, the more specific a user is as to what information he/she wants. For this reason the depth factor, which is used to calculate the weight of nodes is higher for deeper nodes. The formula used to calculate the depth factor has been derived empirically. The condition however is that the depth factor of each depth should be significantly greater than the depth factor of shallower depths. Further experimentation may lead to a more optimal formula. This formula, however, has so far done well in our studies of the system.

The formula for calculating the weight of a node for inclusion in the nodes-visited weights table is as follows:

weight = (number of times visited) x (depth factor)

Table 1 would be the nodes-visited weights table for the example given above.

Table 1. Nodes-visited weights table (example).

Node	Weight
A	5
B	10
C	18.75
D	15.625

A nodes-visited weights table tells us what nodes the user had visited. The weight assigned to a node represents the interest of the user in that node. High weight means high interest.

Shallow nodes represent a broad domain of information (for example, science). Deeper nodes represent more specific information (for example, physics, kinematics, first law of motion). If a user is

looking at a node and then drills down into a deeper node, it implies that he/she is interested in more specific information contained in that deeper node rather than the broad information contained in the shallower node. This is the reason why deeper nodes are assigned higher weights.

3.3. Calculation of Context Overlap Factor (COF)

When a search is submitted, the Knowledge Exchange search engine first selects all those questions from the knowledge base that contain the given keywords. Second, the search engine ranks each of the selected questions according to its relevance to the user's search in terms of context. The most relevant question receives the highest rank.

The relevance is measured in terms of a variable called the *Context Overlap Factor (COF)*, which is calculated for each of the questions. The COF is calculated by determining the overlap between the nodes-visited weights table of the originally asked question and the nodes-visited weights table of the search. High overlap results in a high COF value. The procedure for calculating the COF value is as follows:

1) Initialize the COF value to zero.
2) Create a list of the nodes that are common to both the nodes-visited weights table of the originally asked question and the nodes-visited weights table of the search.
3) For each node in this list, select its weight from the table in which its value is the smallest (if the weight is equal in both tables, then it may be selected from either table). This is the overlap value for that particular node.
4) Add this overlap value to the COF value. When this operation has been performed for each of the nodes in the list, the final COF value has been obtained.

The following example illustrates the calculation of COF (refer to Table 2 for nodes-visited weights table of question and Table 3 for nodes-visited weights table of search).

Table 2. Nodes-visited weights table of question (example).

Node	Weight
A	5
B	10
C	18.75
D	15.625

Table 3. Nodes-visited weights table of search (example).

Node	Weight
A	6
B	15
E	25
F	46.875
G	78.125
H	97.656

Overlap: A: 5, B: 10
➔ COF = 5 + 10 = 15

3.4. Illustration of the search mechanism with the help of an example

We illustrate our context-sensitive search mechanism using the 'Arrays' example given earlier in Section 1.

Suppose that an instance of Knowledge Exchange holds the topics computer programming and telecommunications.

Student A follows the following path prior to asking "What are arrays?"

Information Technology → Computer Programming → Java → Data Structures

Student B however follows the following path prior to asking the same question.

Information Technology → Telecommunications → Wave Propagation Theory → Antennas

The two questions along with their individual nodes-visited weights tables are stored in the knowledge base. The nodes-visited weights table of the first question would be as Table 4 and that of the second question would be as Table 5.

Table 4. Nodes-visited weights table of student A's question.

Node	Weight (formula: times visited x depth factor)
Information Technology	4 (4 x 1)
Computer Programming	7.5 (3 x 2.5)
Java	12.5 (2 x 6.25)
Data Structures	15.625 (1 x 15.625)

Table 5. Nodes-visited weights table of student B's question.

Node	Weight
Information Technology	4 (4 x 1)
Telecommunications	7.5 (3 x 2.5)
Wave Propagation Theory	12.5 (2 x 6.25)
Antennas	15.625 (1 x 15.625)

Let's say that in the future, a third student (Student C) wants to find the question "What are arrays?" in the context of telecommunications. The student may browse the knowledge base looking for the question taking the following path:

Information Technology → Telecommunications → Wireless

The nodes-visited weights table of the search would be as Table 6.

Table 6. Nodes-visited weights table of the search.

Node	Weight
Information Technology	3 (3 x 1)
Telecommunications	5 (2 x 2.5)
Wireless	6.25 (1 x 6.25)

Unable to find the question, the student submits a search using the string "Arrays" or any other string containing the keyword "Arrays". According to a strict keyword match, either stored question would match Student C's search. However, calculating the COF for each of the matched questions reveals a higher relevance value for one of them, the one in the telecommunications hierarchy. The student therefore finds the correct information.

Overlap of search and student A's question (the question with programming as context): Information Technology: 3
➔ COF = 3

Overlap of search and student B's question (the question with telecommunications as context): Information Technology: 3, Telecommunications: 5
➔ COF = 3 + 5 = 8

It should be noted that this mechanism is useful when a user takes a two-step approach to finding information, first browse then search.

4. Conclusion

In this paper we presented a context-sensitive search mechanism that can be used in collaborative help systems. The mechanism considers the recent activity of a user as the context of their questions and searches. With the help of examples we have demonstrated how our context-sensitive search mechanism can be advantageous over search mechanisms that are not context-sensitive.

5. Acknowledgments

This research was carried out at Drexel University, PA, USA and was sponsored in part by National Science Foundation (NSF) (Award DUE-0085713).

6. References

[1] Ackerman, M., and Malone, T.W., "Answer Garden: A Tool for Growing Organizational Memory", *In Proceedings of the ACM Conference on Office Automation Systems*, New York: ACM Press, 1990, pp. 31-39.

[2] Ackerman, M., and McDonald, D., "Answer Garden 2: Merging Organizational Memory with Collaborative Help", *In Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, New York: ACM Press, 1996, pp. 97-105.

[3] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley Professional, 1999.

[4] Chaiklin, S., and Lave, J., *Understanding Practice: Perspectives on Activity and Context*, Cambridge: Cambridge Press, 1996.

[5] Dey, A.K., and Abowd, G.D., "Towards a Better Understanding of Context and Context-Awareness", *Technical report GIT-GVU-99-32*, College of Computing, Georgia Institute of Technology, 1999.

[6] Dumas, S.T., Cutrell, E., and Chen, H., "Bringing Order to the Web: Optimizing Search by Showing Results in Context", *In Proceedings of the CHI'2001 Conference on Human Factors in Computing Systems*, New York: ACM Press, 2001, pp. 277-283.

[7] Experts Exchange, retrieved March 3, 2004 from http://www.experts-exchange.com/.

[8] Google Answers, retrieved April 29, 2004, from http://answers.google.com/answers/.

[9] Google, retrieved April 29, 2004, http://www.google.com/.

[10] Horton, M., and Adams, R., "Request for Comments (RFC) 1036 – Standard for Interchange of USENET Messages", *Network Working Group*, 1987.

[11] Klemke, R., "Context Framework: An Open Approach to Enhance Organizational Memory Systems with Context Modeling Techniques", *In Proceedings of the Third International Conference on Practical Aspects of Knowledge Management (PAKM 2000)*, Basel, Switzerland, October 2000.

[12] Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B., and Karger, D.R., "The Role of Context in Question Answering Systems", *In Extended Abstracts of the CHI'2003 Conference on Human Factors in Computing Systems*, New York: ACM Press, 2003, pp. 1006-1007.

[13] Math Forum, retrieved March 3, 2004, from http://www.mathforum.org/.

[14] Matwin, S. and Kubat, M., "The Role of Context in Concept Learning", *In Proceedings of the 13th International Conference on Machine Learning*, Bari, Italy, 1996.