ASKME: Adaptive and Self-evolving Knowledge-base for Mobile Environments

Hassan Artail Jad El Hage
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
hartail@aub.edu.lb jge05@aub.edu.lb

Reda Aouad Kassem Fawaz
Department of Electrical and Computer Engineering
American University of Beirut
Beirut, Lebanon
rma66@aub.edu.lb kmf04@aub.edu.lb

Abstract

In spite of the existence of several systems that organize and offer information to users (e.g. Internet, Intranet, or private databases) finding the desired data is still a time consuming task. ASKME solves this problem by providing users with a collaborative learning environment which evolves through direct and indirect user contributions. The system includes a credit/debit system to make sure users participate in providing answers and feedback. The system can also provide users with online material that it has located. The current system is implemented as a web server and thus the knowledge-base (KB) is centralized. Future plans include allowing for distributed data.

1. Introduction

Information is no longer available only through monopolized systems. In higher learning for example, it is no longer sufficient for students to depend on class notes, handouts, and assigned books to acquire a thorough understanding of subjects and apply it in laboratory experiments and projects. Computer users are increasingly depending on electronic systems, such as the Internet and Intranet, for information resources. In spite of all the facilities provided to speed up search and looking up information, finding the right data is still a time consuming task. Having a system that lets users learn from each other and acquire focused information on their behalf can prove valuable in today's demanding environments. This is especially important for organizations, where collaboration and knowledge sharing have been recognized as key enablers for gaining organizational effectiveness and competitive advantage [1, 2]. A roadblock, however, has been the low level of user acceptance and the fact that current shareware systems require effort to share knowledge. A system that does this on behalf of users can make knowledge sharing a seamless process.

2. Solution

2.1 Proposed Solution

Our proposed system consists of a knowledgebase (KB) for collaborative learning environments, which evolves through direct and indirect user contributions. These contributions can be in the form of answers to questions, deposit of material (documents, links, etc.), or guidance that helps the system search for and retrieve material from the Internet or other sources behind the scene and on its own. The system embeds intelligence aspects, which enable it to analyze and process information into knowledge. In learning institutions, students and even instructors can tap into this knowledge to get the needed assistance and gain awareness about their environment and relevant technologies. A system prototype was implemented on top of a dynamic network of mobile devices through which users access the system and interact with one another. Through their mobile units, users can directly assist others through answers and documents. Possible future additions include implementing a distributed version of the KB and adding caching modules on the dispersed mobile units. These additions will enable the system to adapt better to mobile environments where access to the KB is not always possible. In short, the system would model a community within which members work cooperatively toward acquiring, organizing, and disseminating relevant knowledge.

2.2 Previous and Ongoing Work

The last few years have witnessed a rapid advancement in Mobile and Wireless technologies, which has triggered a considerable number of initiatives for introducing new paradigms in learning. Here, we briefly go over a number of related projects and technologies that relate to our proposed work.

2.2.1 Mobile Learning Projects. The European-led Mobilearn project aims at defining models for teaching and learning in a mobile environment in addition to developing a mobile learning architecture [5, 7]. This project focuses on the interaction between the mobile user and the content server, and falls short of creating a collaborative dynamic environment in

This work was funded by a grant from Mr. George Kadifa .

which mobile users act as a community to facilitate information exchange.

Another m-learning initiative, *ActiveClass*, tries to integrate mobile devices into the classroom environment to create a "virtual student" [3]. The main objective is to facilitate student-teacher interactions whereby students can pose questions anonymously and teachers can answer or comment on them during the same class or at a later time.

The *Samsara* project at the University of Michigan, tries to achieve fairness in peer-to-peer storage systems to ensure that nodes consume no more resources than what they contribute [8]. The interesting aspect of this project is that it tries to impose a system of punishment and reward in a distributed mobile network and shares this concept with our proposed framework. The reliability and fairness of this system is not made clear, though. There is no central authority for credit exchange that is in place to supervise these operations and record each node's contribution and usage.

The MIT *OpenCourseWare* and *iCampus* projects address online course material publication and access to remote services and data sources. The plan is to solve communication problems between students and instructors in large classrooms through the use of mobile devices.

The *Sharable Courseware Object Reference Model* (*SCORM*) is a suite of technical standards that enable web-based learning systems to find, import, share, reuse, and export learning content in a standardized way [11]. Of special importance to our proposed work is the Content Aggregation Model (CAM) part that defines XML meta-tags for describing learning content, binding it, and packaging it.

2.2.2 Network Infrastructure. Mobile Mesh Networks [15] will be considered as a solution for interconnecting the mobile devices in our system due to their ability to extend network reach. A wireless mesh network is a multi-hop system in which devices assist each other in transmitting packets through the network. To achieve adequate connectivity it might be necessary to require equipment to be on continuously, which would drain battery life quickly. This is one of the basic limitations of this technology since device owners may be reluctant to allow their handhelds to act as routers for others without some kind of a return. An equally important issue to resolve relates to covertness, as users might not want their devices to be transmitting without their direct control.

Wireless ad-hoc networks represent another option for the underlying network infrastructure of our proposed work due to its properties in resolving communication problems in areas with little or no infrastructure. Each node within the network can operate as a router or a host depending on the event. Several issues exist, however, that are associated with such networks, mostly related to the volume of broadcast and energy consumption. Although such subjects are not the focus of our work at this stage, but we intend to experiment with some of the proposed approaches in this regard and exploit the work that has been done on mobile ad hoc networks, mostly in relationship to routing (e.g., AODV and OLSR [7]).

3. System Description

3.1 General System Description

The knowledgebase includes intelligence aspects for maximizing the probability of providing users with the right information at the right time, virtually anywhere within a prescribed geographical area. The inputs to the system are the users' questions and answers, while the outputs are the question-specific answers in addition to relevant online material. In one of the deployment scenarios, access to the system will mainly be through Personal Digital Assistants that communicate via a wireless network such as a Mobile Ad-hoc Network (MANET). Figure 1 shows a generalized schematic aimed at outlining the different system components and their functionalities.

3.1.1 The warehouse. It is the central component, coordinates all of the system activity.

3.1.2 The sentence similarity engine. It is the most involved component, is based on Natural Language Processing (NLP), and is able to analyze and understand user questions and identify questions of equivalent meaning. One of the principle functions of this component is to map syntactically-different but semantically-same questions to the same answer, which can be returned to the user.

3.1.3 The browser module. It consists of a web-based interface through which the user can interact with the system and receives the answers to his or her questions.

3.1.4 The web-searcher (crawler) module. It is responsible for searching the web for answers to questions which could not be matched in the KB. This component supports HTML pages as well as other standard formats, like PDF, DOC, and PPT.

3.1.5 The system databases. Two databases are maintained: the 'Data' database which holds the knowledgebase that stores the question-answer pairs (QAPs) as well as their related and support information, and the 'Record' database which contains the users' information, including identification information as well as credit/debit status.

Next, we will detail the implementation and capability of each component and describe how the components interact with one another to provide the overall functionality.

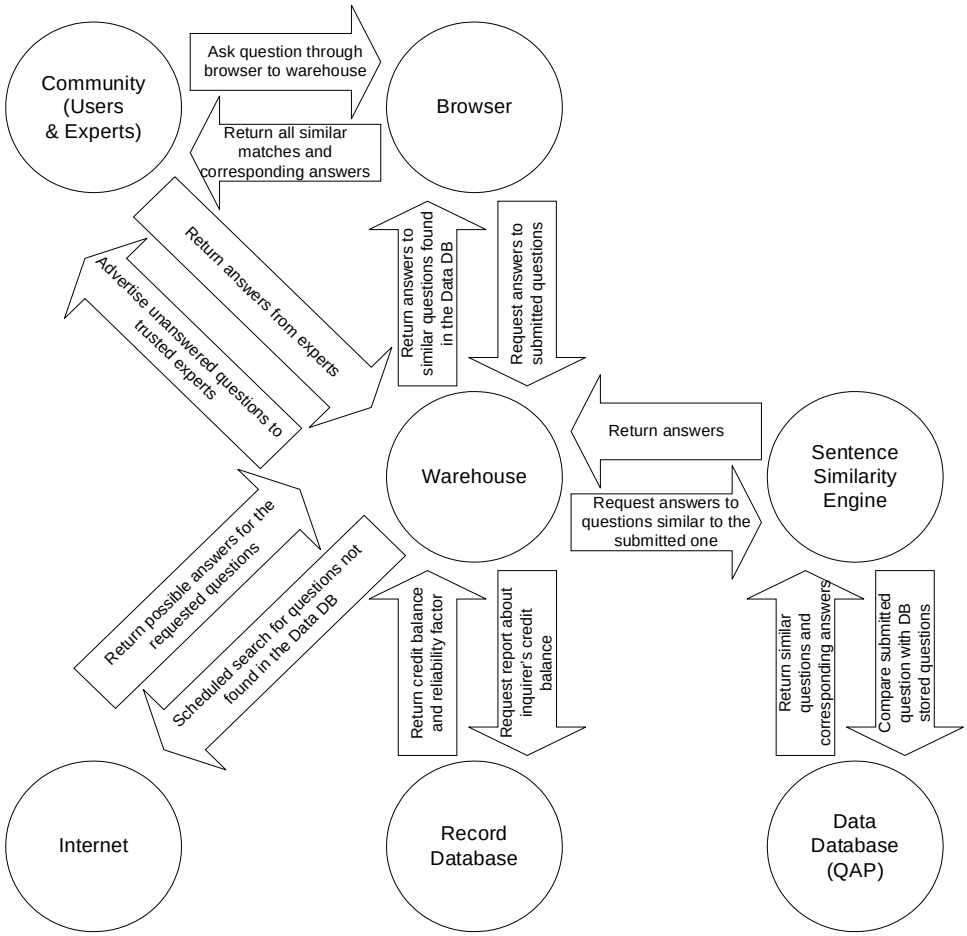

Figure 1. A General Diagram of the System.

3.2 The Warehouse

As stated earlier, the warehouse has the job of coordinating the system activity. This includes routing every question through the database, as well as recording and processing user ratings and managing the credit/debit system.

3.2.1 Routing Questions. When submitted, a user's question follows an intricate journey throughout the system's Data database tables. It is up to the warehouse to guide this journey. The diagram in Figure 2 details a question's path through the system.

When a question is first asked, it is first forwarded to the sentence similarity engine which will attempt to find an equivalent KB question. That is, if an equivalent question is found, then certainly an answer exist.

If an equivalent question is found, the answers to the top 3 matching KB questions are presented to the asking user who can in turn rate the answer's relevance to his or her question. This rating can then be used to statistically assert the system performance and fine-tune system parameters, such as the question matching threshold.

In case an equivalent question is not found, and according to the asking user's initial choice, the question can be posted for other users to answer, or the system can try to find an answer via web searching.

Apart from being presented to the asking user, these non-KB answers are made subject to user ratings. In case the ratings meet certain criteria (number and quality of ratings) a new QAP is formed and inserted into the KB. As such, the KB self-evolves.

Asking users are asked to specify when question expire (a huge value can be entered for non expiring questions). Periodically, the warehouse deletes expired questions from the KB to rid it of time-bounded questions.

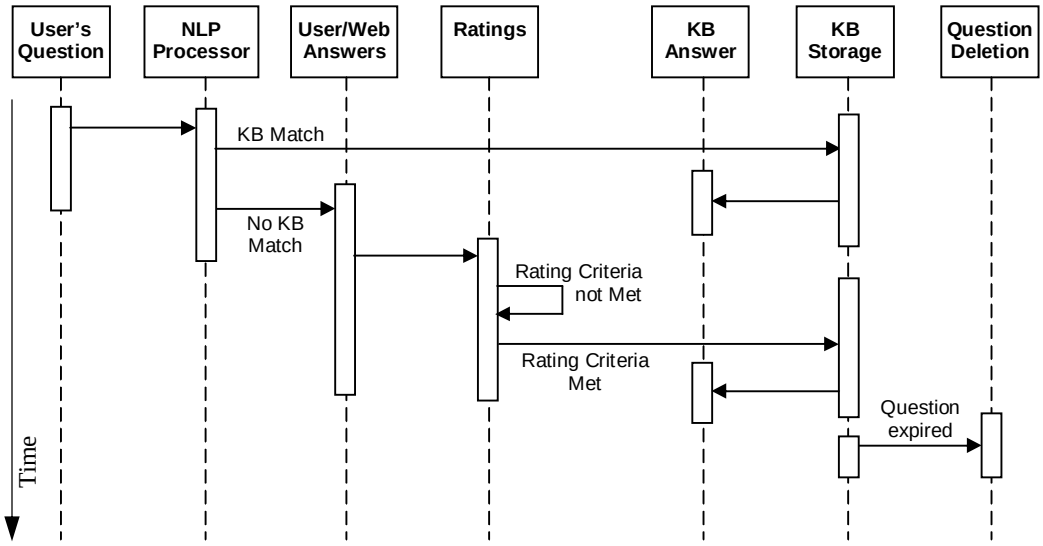

Figure 2. High level question processing through system.

3.2.2 Processing User Ratings and Managing Credit/Debits. To ensure that a user submits high-quality answers, a credit system is implemented; it is intended to have users contribute toward growing the data database. In order to achieve this goal, the credit points can increase or decrease according to the contribution of the user. The credit system is described as follows: first, any user who signs up in the system is granted a given number of credits (e.g., 100). This allows him or her to ask several questions since he or she is not considered a contributor to the system at first. Once he or she starts asking questions, credits begin getting deducted from his account, preventing him or her from asking indefinitely without contributions to the system.

As was mentioned, in order to get credits, the user must contribute to the system either by proposing answers to others' questions, or by reviewing and rating other's proposed answers. When a user is reviewing a proposed answer to his/her own question, he/she has the option to rate the answer, from 1 to 10, 10 being the highest rating. This procedure gives the submitter of that particular answer credits whose value is equal to the awarded rating divided by two. The justification behind this is that the answer provider is contributing to providing answers that would eventually be part of the KB.

Credits are also realized when users rate other users' answers to the questions of other users. This process however does not give the user any credits at first, but when an answer is rated with more than 10 times and the average rating is higher than 6/10, each user who rated that question gets credits according to how close his or her rating was to the average. That is, the closer his or her rating the higher the credits he or she obtains; a maximum of 8 credits can be gained in case the user's rating was equal to the average of all ratings.

The system differentiates between three types of users: regular users, expert users, and administrators (we will discuss administrators later). Expert users benefit from infinite credit and reliability (their answers are not subjected to ratings) as well as from the ability to trash an inadequately proposed answer: if an expert user is reviewing proposed answers and finds out that one answer is not valid for the question asked, he or she can trash the answer and the user who proposed it loses all his accumulated credits and gets notified of the event once he or she logs in. The only way for him or her to regain credits is to start proposing new answers or ratings to previously proposed ones. This mechanism serves as a gateway meant to prevent as much as possible attempts to sabotaging or tricking the system by giving careless and totally invalid answers. It therefore ensures that users in the system provide contributing answers that add more knowledge to the KB and help its growth.

3.3 Sentence Similarity Engine

This engine is at the core of our system: it has the major task of generating similarity scores between two input sentences. This sentence similarity is used to match incoming questions to question-answer pairs in the KB. In order to compare the two input sentences and generate a similarity score, a series of preprocessing steps must be performed on the sentences. Hence, the task of the sentence similarity engine is achieved using a two-step process which starts off with a sentence preprocessing procedure, followed by a *Scoring* phase. This is shown in the diagram of Figure 3.

3.3.1 Preprocessing input sentences. This step takes an input sentence as a string and outputs an array of WordInfo objects which consists of a word and its associated part-of-speech and semantic information.

It is important to note that this step can be performed on each input sentence individually. This fact will be benefited from by storing the WordInfo arrays of KB questions directly

in the KB such that this preprocessing is only performed on newly entered questions.

The preprocessing is done in four stages. First, the input question is tokenized, that is the single string of characters is divided into an array of individual tokens, each token representing a word (or number or punctuation) in the original string. Then, the array of tokens is tagged, meaning that each token is assigned a part-of-speech tag. Next, the sense of each individual token is determined as defined in the *WordNet.Net* library [19] using a word-sense disambiguation procedure. Finally, tokens are filtered, and only "relevant" tokens are maintained. The following discussion details each stage:

3.3.2 Tokenizing. This stage is accomplished using a regular expression tokenizer implemented using Python and the Natural Language Tool Kit (NLTK) [20]. The regular expression used distinguishes between words, numbers and punctuation and separates each into individual tokens that get arranged into an array and forwarded to the tagger.

3.3.3 Tagging. Part-Of-Speech tags are assigned using a tagger process implemented in Python using NLTK. The tagger uses a 5^{th} order tagger and is trained using the NLTK provided corpora which contain large amounts of tagged text. The corpora used are the Brown and the Penn Treebank corpuses containing different genres of tagged text.

As a 5^{th} order tagger, training the tagger is both time and memory consuming since, for every encountered word and its associated tag, the parts of speech of the 5 preceding words are taken into consideration. This entry, consisting of a word, its tag, and the tags of the preceding words, is used to update the tagger table. Once training is complete, this table consists of a probability table which can be used to assign a tag to a word given the tags of up to 5 preceding words. Given, the array of tokens from the tokenizer, the tagger simply starts tagging the words or tokens one after the other in a sliding window fashion.

The different parts-of-speech that are used contain but are not limited to: Noun, Verb, Adjective, and Adverb.

Since the tokenizer and tagger processes are implemented in Python, and since the remainder of the system is written in C#, interfacing between these two processes and the system was implemented using piping, i.e. re-routing the standard input and output streams of the python program to the C# program.

3.3.4 Word-sense disambiguation. In order to obtain a better insight into the meaning of the input sentence, and since some words may have different senses even when having the same part-of-speech, determining the specific sense of the words in the input sentence was necessary. In order to do this, a process called word-sense disambiguation was used.

First, a lexicon had to be selected and integrated into the system. WordNet is a database of over a hundred thousand words with meanings and a complex architecture of word links. WordNet.Net is a .NET framework library for WordNet and can be easily integrated into a C# project.

The sense disambiguation process uses a context-based approach and uses a sliding window algorithm. This means that it uses the context equivalent to the words in a window of given size around the currently processed word to determine the sense. What it does is that it looks up the definitions of the multiple senses of the current word given its part-of-speech. Next, it will search for occurrences of context words (or their synonyms) inside these definitions. The sense whose definition has the greatest amount of context word hits is selected to be the sense of the currently processed word. Each word in the sentence is processed similarly until all word senses have been determined.

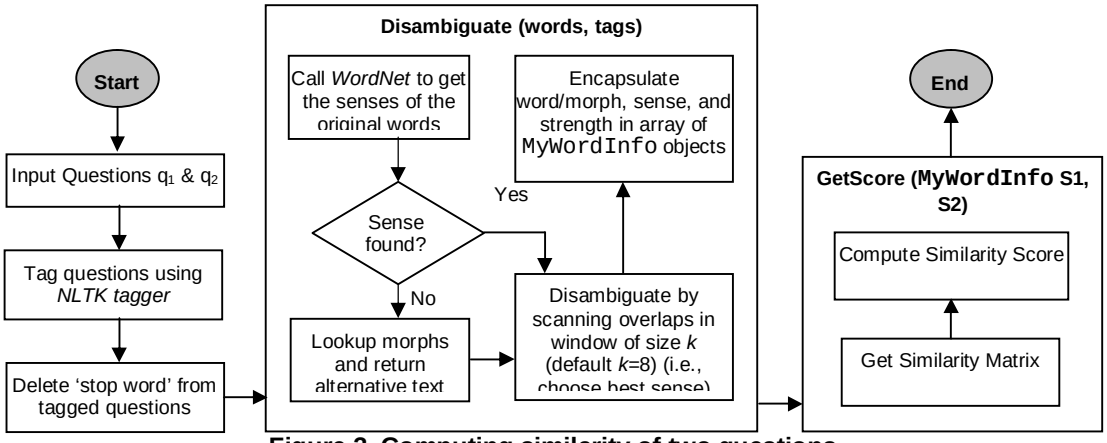

Figure 3. Computing similarity of two questions

3.3.5 Stop-words filtering. Some words which do not provide any meaningful information concerning the question's content have been found to induce unwanted results such as excessively high scores on questions which do not match. A few examples of such words are the articles "a", "an", and "the". In order to solve this problem, these unwanted words,

called stop-words are filtered out and removed from the WordInfo array representing the sentence. The filtering is implemented by simply checking the words in the WordInfo array against a list of predefined stop-words. Note that this step is left to the end in order not to change the context of any word which could affect the tagging and sense disambiguation procedures.

3.3.6 Scoring. The final similarity score is a float value ranging between 0 and 1. This value is calculated by first computing a similarity matrix which contains the similarity of every word in one question to all other words in the second question. The similarity between two words is taken as an inverse relation to the Word Distance separating these two words (with their respective parts-of-speech and senses) in the WordNet lexicon. Having computed these word-to-word similarities, the similarity between the two sentences is computed as an average of these individual similarity scores: It is taken to be the sum of the similarity scores of matching words in the two sentences divided by the total number of words in both sentences. Two questions are considered to match if their similarity score is greater than 0.6 – this number was obtained after rigorous testing.

3.3.7 Searching. The KB holds a multitude of question-answer pairs (QAPs) with their preprocessed `WordInfo` objects. While a user submits a new question, he/she specifies the topic of the question from a defined list of specific topics. Once the question is submitted, the `WordInfo` objects array of the new question is computed and scored against all the questions in the same topic whose `WordInfo` objects are simply loaded from the database. This speeds up search time by storing `WordInfo` data in the database and avoiding re-computation and also by searching only among KB entries with same topic.

3.4 Browser

The Browser is a web interface that enables users to benefit from all of the system's functionalities. Users can navigate to the site from any machine (PC or PDA) connected to the network. As such the system is accessible at all times without the need to install software on individual machines. The interface was written in ASP.NET and benefits from all the latest software technologies. Figures 4 and 5 provide parts of the two screens through which users can ask the questions and specify the domains, and view the status of submitted questions, respectively.

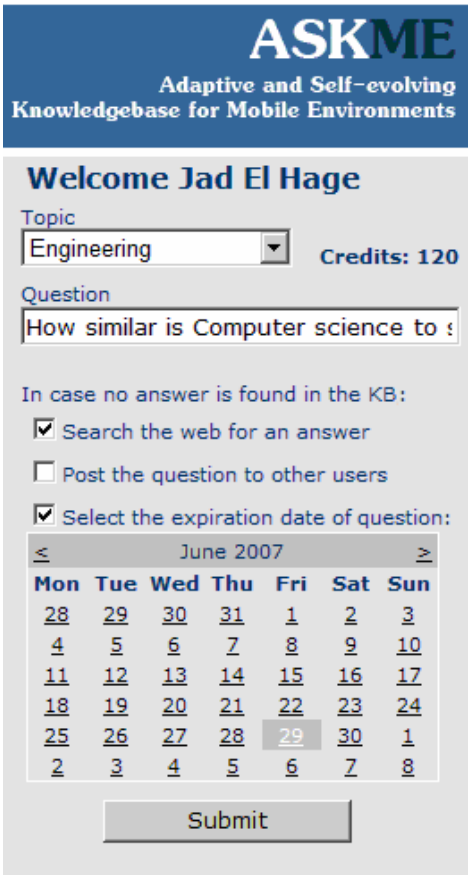

Figure 4. Part of Question Asking Screen.

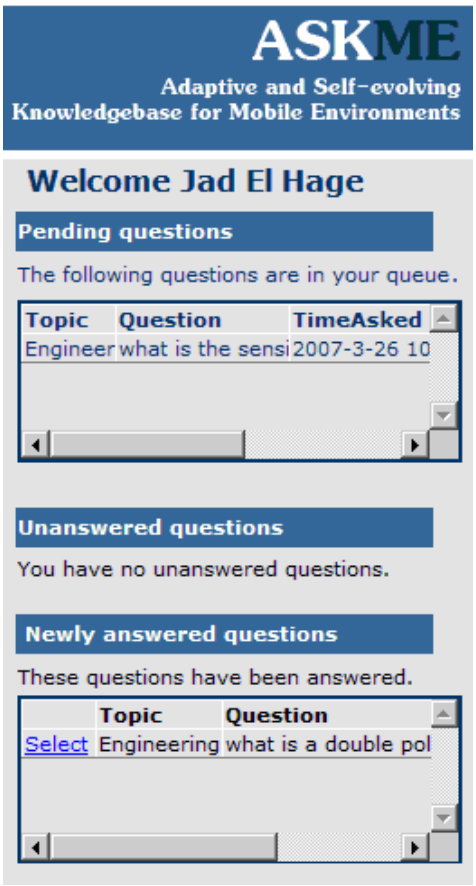

Figure 5. Part of Question Status Viewing Screen.

3.4.1 User Credentials and Security

In order to benefit from the system, a user must first log into his/her private account using username and password. Accounts can be assigned on demand or self-created using an available sign-up form.

In order to prevent from simple URL copy hack, when a user logs in, a unique session is created with a maximum idle period of 10 minutes. The session existence is checked at every page load, in case the session is missing, the user is redirected to the log-in page. The session is destroyed at log-out.

Also measures have been taken to prevent from SQL injection when asking a question or proposing an answer. Only properly formatted sentences are accepted.

3.4.2 System Functionality. There are three different types of users:

Regular users. They can ask questions, propose and rate others answers; but their proposed answers are submitted for rating. They can also view the status of their pending questions as well as their archived question-answer pairs.

Expert users. They have the same abilities as regular users but their proposed answers are not subject to rating, and are automatically accepted. An expert can also trash a user's proposal. It should be noted that this group of people are supposed to be knowledgeable in their areas and have interest in having the system function properly. For example, in a university setting, experts can be professors or trusted graduate assistants.

Administrators. They have the same abilities as expert users, but they also have access to administrative functions such as changing a user's type or credits, accessing the database. In most cases, these would be technical people involved in programming the system.

When asking a question, users can specify the topic of the question; this is used to enhance searching by limiting the search space to specific topics. Users can propose new topics to administrators. Also, users specify their choice of secondary source in case no answer was found in the KB. They can choose to submit the question to peers, or search the web for answers, or both. Finally, users can specify if their question expires, and in case it does, the expiry date. These details are stored with the question in the database.

Users can propose answers to unanswered questions that were specified to be submitted to users. The proposing user can view all unanswered questions according to their topic. A question can be selected and a proposal submitted. Once a proposal is submitted, the question is taken out of the list of questions.

Users can rate others' answers to questions. Users select a topic and a question from that topic and are able to review the proposed answers and rate them. All of the user's pending questions and newly answered questions are made available to the user through a check status page. Once newly answered questions are viewed, they are stored in an archive table for later review.

3.5 Web-Searcher

In case no equivalent question was found in the KB, and in case the asking user specified to get answers from the web, the web searcher module takes control over the question: it searches using the Google web crawler for potential answers and provides the top 5 pages – or documents – returned by Google along with their descriptions and links. This is accomplished through a web service provided by Google, which allows us to do an automated search for any query. Thus, the question is given to that service as asked by the user, and the top results are given back to the user as "WEB" answers to his unanswered question. That module was easy to implement, since Google's web service is provided as an API that can easily be used in the .NET environment. This module supports four types of web results provided by Google:

- Regular HTML pages
- Microsoft Office Word documents
- Microsoft Office PowerPoint documents
- Adobe Acrobat PDF documents

Whatever the document type is, each one of the 5 top results is processed and the relevant data is extracted using specific routines written for each of the four document types. This allows the system to return concise answers or descriptions to the user. For example, considering a Microsoft Office Word document, the paragraph containing the greatest number of keywords from the question is summarized and returned to the user.

4. System Performance

4.1 Matching Speed

The matching procedure is the bottle-neck of the system since a new question has to be matched against all the QAPs in the KB and their number can grow indefinitely. To speed up processing, three approaches were taken: a simple technique that was employed to speed up the search was to categorize the questions according to their topics and thus allowed for limiting the search space to a subset of the KB. As a result, more specific topics render more restrained search and faster search. Another technique that we implemented in order to decrease computation time and eliminate repetitive computations, was to computer the `WordInfo` array of every question only once and storing it in the database. Finally, since the system has many different tasks to complete, we have tried to benefit from parallelism as much as possible by separating the different modules into separate processes and/or threads.

With the above enhancements, and to give an idea, under a regular load, the average time it takes the system to search through a series of 200 questions is about 10 seconds. This number is encouraging, but nevertheless, we will look for additional ways to decreasing it in the future.

4.2 System Reliability and Matching Accuracy

This testing concerned getting results about the accuracy of the system regarding questions matching. A set of 200 questions was carefully prepared with corresponding answers and fed into the QAPs table, so as to have the system consider them as part of the KB. For each of those questions, five similar versions were asked: similar either in the sentence structure or in the keywords that it contains. The versions were obtained by asking the question in different forms, changing words with others having similar meanings, using abbreviations, changing the tense, etc. We implemented a user interface that shows the three top most matching questions in the KB to the posed one along with associated similarity scores. For each submitted question, manual inspection of the questions was performed to obtain the number of questions in the KB that actually matched the asked question. The results are summarized in Table 1 and in the pie chart of Figure 6.

Table 1. Testing Results

Matching Questions	Non-Matching Questions	Average Score	Score Standard Deviation
84 %	16 %	0.795	0.212

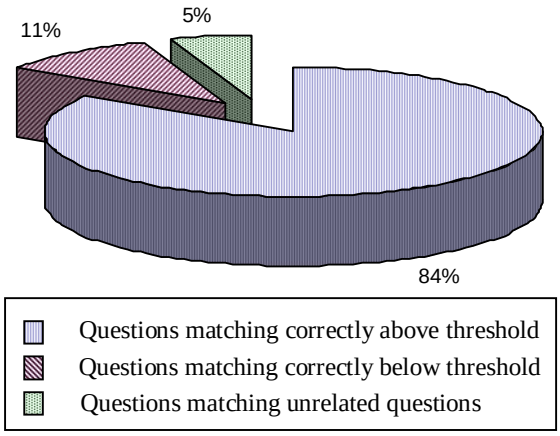

Figure 6. Matching Results based on a test of 200 Questions.

The tests were also used to measure the times in answering each question. To gain a detailed sense of the response time of the system, the times were sorted against the number of words in the question (excluding the non-stop words). Figures 7 and 8 show the results of the disambiguator and comparator stages of the system through a scatter plot.

5. Future Work

Several ideas for improving the performance of the system and its usability have been compiled. First, concerning current performance of the search engine, it may be regarded as satisfactory; however the engine remains the bottleneck of the system. We believe that it can be further enhanced. One such enhancement would be to add more indexing in the databases to improve the matching process.

One feature that would improve the usability and also the accuracy of the system is a spell checker, to check the correctness of user questions before submission. One possible solution that has already been thought of is to use Microsoft Word's integrated spell checker.

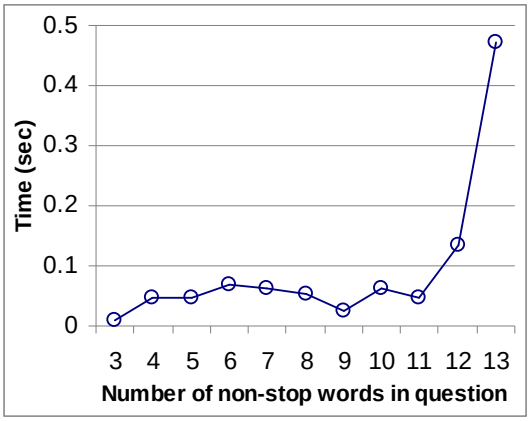

Figure 7. Processing time of Disambiguator.

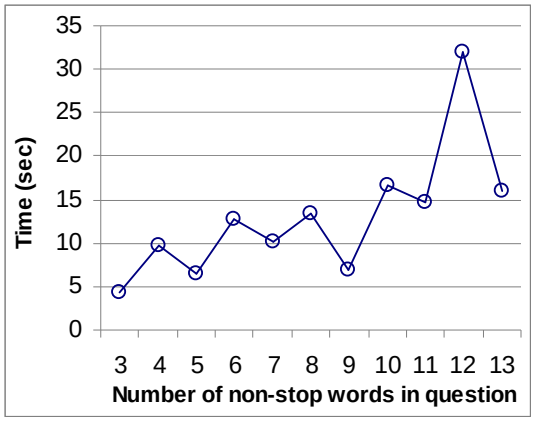

Figure 8. Processing time of Comparator.

Acknowledgment

The authors would like to thank Mr. George Kadifa for his generous grant, which financed this research and made it possible to develop the ASKME system. We also thank the head of the Faculty of Engineering and Architecture at the American University of Beirut, Dean Ibrahim Hajj, for his efforts to acquire this grant.

References

[1] J.R. Riesenberger, Executive Insights: Knowledge – The source of sustainable competitive advantage. *Journal of International Marketing*. 6(3), 1998, pp. 94-107.

[2] K.M. Wigg, Introducing knowledge management into the enterprise. In *Knowledge Management Handbook*. J. Liebowitz (ed.), Boca Raton: CRC Press, 1999.

[3] http://www-cse.ucsd.edu/~wgg/Abstracts/activeclass-cscl03.pdf (THE ACTIVECLASS PROJECT: EXPERIMENTS IN ENCOURAGING CLASSROOM PARTICIPATION)

[4] http://www.eee.bham.ac.uk/sharplem/Papers/ijceell.pdf (Disruptive Devices: Mobile Technology for Conversational Learning)

[5] http://www.mobilearn.org/download/results/guidelines.pdf ("Guidelines for Developing Mobile Learning Deliverable" (pdf file - 221 KB))

[6] http://www.eee.bham.ac.uk/sharplem/Papers/handler%20cal2001.pdf (A Systems Architecture for Handheld Learning Resources)

[7] http://www.mobilearn.org/download/results/Mlearn_paper.pdf (A Task-Centered Approach to Evaluating a Mobile Learning Environment for Pedagogical Soundness)

[8] http://mobility.eecs.umich.edu/papers/sosp03.pdf (Samsara: Honor Among Thieves in Peer-to-Peer Storage)

[9] http://www.eee.bham.ac.uk/sharplem/Papers/mobile%20learning%20puc.pdf (The Design and Implementation of a Mobile Learning Resource)

[10] http://www.m-learning.org/docs/the_use_of_palmtop_computers_for_learning_sept03.pdf (the_use_of_palmtop_computers_for_learning)

[11] http://www.adlnet.org/screens/shares/dsp_displayfile.cfm?fileid=481 (SCORM refernce model overview)

[12] http://download.macromedia.com/pub/solutions/downloads/elearning/scorm_flashlo.pdf (Making a Macromedia Flash MX Learning Object SCORM-Conformant)

[13] http://ltsc.ieee.org/wg1/files/IEEE_1484_01_D09_LTSA.pdf (IEEE Draft Standard for Learning Technology — Learning Technology Systems Architecture (LTSA))

[14] www.elearningforum.com/meetings/ 2003/january/Rjs-jan-2003.pdf (CISCO's e-learning architecture - presentation)

[15] www.elearningforum.com/meetings/2003/january/eLearningForum_MS2.pdf (Creating a common learning infrastructure - Microsoft - presentation)

[16] http://www.itu.int/osg/spu/newslog/categories/meshNetworks/

[17] http://en2.wikipedia.org/wiki/ad_hoc_protocols_implementations

[18] Knowledge Discovery Using KNOW-IT (KNOWledge base Information Tools)

[19] WordNet.Net: http://www.ebswift.com/OpenSource/WordNet.Net/

[20] NLTK: http://nltk.sourceforge.net/index.php/Main_Page