NJBAS

Available online at http://ajol.info/index.php/njbas/index ISSN 0794-5698
Nigerian Journal of Basic and Applied Science (2010), 18(2): 260-268

A Textual Case-Based Mobile Phone Diagnosis Support System

***A. Almu and K. M. Maiyama**

Dept. of Mathematics, Computer Science Unit ,Usmanu Danfodiyo University, P.M.B 2346, Sokoto – Nigeria
[*Author of correspondence: almul2003@yahoo.com]

ABSTRACT: Java Cases and Ontology Libraries Integration for Building Reasoning Infrastructures (jCOLIBRI) is a framework which makes the development of Textual Case-Based Reasoning (CBR) applications easier by providing the preprocessing of text methods, textual similarity methods and appropriate representation for textual cases which are the major techniques needed in any CBR systems. In this paper, a Mobile Phone Diagnosis Support System is presented as an extension to jCOLIBRI which accepts a problem and reasons with cases to provide a solution related to a new given problem. Experimental evaluation using some set of problems shows that the developed system predicts the solution that is relatively closer to the user given mobile phone problem. The solution also provide the user valuable advise on how to go about solving the new problem.
Keywords: Textual case-based reasoning, jCOLIBRI

INTRODUCTION

Mobile Phone users in Nigeria especially the new users experienced some common similar phone problems such as keypad not responding repeatedly over the years and the solutions to these problems are addressed by the repairer's of the phones. Since the number of phone users are increasing drastically and also the same problems has been taken again and again to the repairers, there is a need to provide a system that would be able to store and index these similar past textual cases and then provide their solutions to the users when needed. Moreover, whenever a repairer submits a similar problem, its corresponding solution can be reused and presented to the repairer. This is achieved by applying Textual Case-Based Reasoning techniques to predict the outcomes of new cases or suggest solutions about the problem at hand. These promising techniques can mimic a human being's typical way of reasoning when solving a new problem by searching through a collection of previous cases and use similar cases to solve the problem.

METHODOLOGY

Case-Based Reasoning (Bergmann *et al*., 2005) is an artificial intelligence methodology that solves problems by using previous information with a view to providing solution to a given problem. A case is described as a record or collection of a past experiences or problems together with solutions or outcomes to be used by the system while solving a given problem (Pal and Shiu, 2004). CBR compares a new problem or a case to its previously solved cases which makes it easier for the system to draw inferences and make decision about a given case (Weber *et al*., 2006). The problem solving using this approach enables the CBR system to solve problems or cases even if they are not completely similar to its previous experiences. This is what makes CBR problem solving approach more advantageous than other problem solving approaches such as rule-based and information retrieval (Lenz, 1998).

Textual Case-Based Reasoning (TCBR) (Weber *et al*., 2006) is a subfield of CBR problem solving methodology that deals with the research and implementation of case-based reasoning using the knowledge sources that are in textual form such as Frequently Asked Questions (FAQs). TCBR is particularly

aimed to use a specific domain that involves textual cases in an automated or semi-automated way to support problem solving by using case comparison in order to derive relevant solution(s) to a given problem (Weber *et al*., 2006). The basic idea behind Textual CBR is that, it uses textual documents as cases and compares these cases in terms of similarity with a given query in an attempt to retrieve similar documents that are likely to be useful for responding to a query as a solution(s). The similarity of cases to be retrieved does not rely merely on keywords matching alone but also consider the similarity measure constructed during the knowledge acquisition process (Lenz *et al*., 1998). Both CBR and TCBR used the four-step processes of CBR life cycle namely retrieve, reuse, revise and retain as an approach to problem solving in any given domain. Unlike CBR which deals with different sources, the TCBR focus on dealing with specific domain or problem where the sources are in texts form (i.e. textual in nature). This is what makes it necessary to use the TCBR approach for the development of mobile phone diagnosis support system since all the problems are in textual form.

jCOLIBRI is a free and open source object oriented framework developed in java programming language for developing Case-Based Reasoning systems (Recio-Garcia *et al*., 2008). jCOLIBRI framework is reusable and extensible in the sense that, the developer can easily reuse its source codes and even extend its classes to build a new system. Similarly, some of the classes can be simply implemented as abstract classes by the developer. This makes it possible to be considered and used in developing this Textual Case-Based mobile phone diagnosis support application. Another benefit of adopting jCOLIBRI in this work is its ability to ease the implementation of Textual CBR system by providing the preprocessing of text (i.e. tokenization and stop words filtration) methods, textual similarity methods to ease the retrieval and also appropriate representation for textual cases. jCOLIBRI also comprises of some tools with a view to improving the performance of the system such as Lucene search engine to provide the functionalities of indexing and searching to the diagnosis support system.

System Design: The Design Methodology for Diagnosis Support System comprises of different parts that works together for the purpose of diagnosing the users' mobile phone given problems. It consists mainly of four (4) parts which includes the User Interface, Problem Preprocessing, Search Engine and the Case Base. The overall structure of the application is shown in Figure 1. The problem is to be entered through the system interface by the user and when the system diagnoses the given problem its feedback would be also displayed on the interface as the diagnoses results for the user to see. The processing of problem text into a form suitable for the search engine to use is handled by the problem preprocessing component. The search engine compares the user entered problem with the previous cases contained in the case base and retrieved those solutions that are relevant to the user given problem. The application case base stores the cases (problems and solutions) to be used by the system when diagnosing a user given problem.

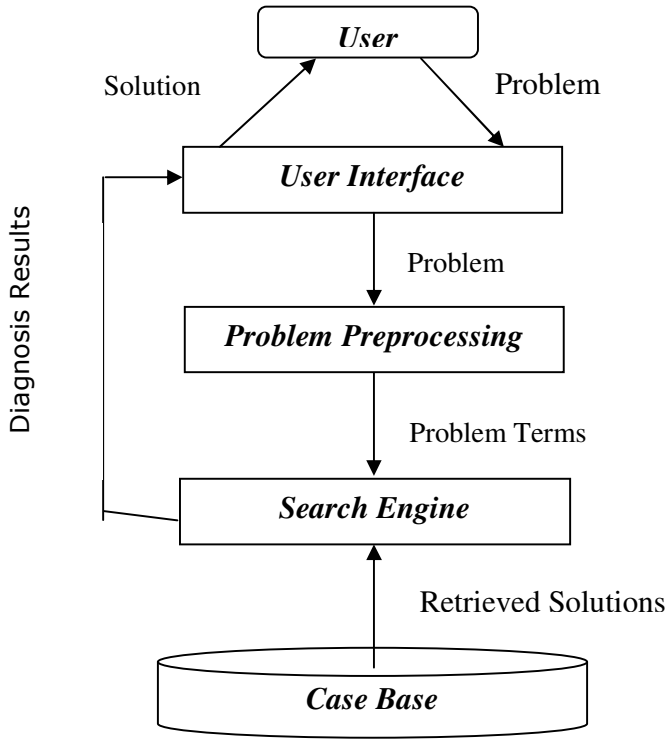

Figure 1: The Architecture of the Mobile Phone Diagnosis Support System

User Interface: The User Interface is designed specifically to ease the means of interaction between the user and the diagnosis support system. According to Ghowdhury (2004) the two major functions of the user interface in a typical Information Retrieval (IR) system includes allowing the users to search for the information needed and display relevant search results, and also to make it simpler for users to carryout various tasks such as saving results, modifying the query and so on.

The system interface is a medium through which a user inputs a problem to the system and click on diagnosis button which triggers the necessary communication between the system's components with a view to diagnosing the problem. The result for the inputted problem is displayed via the interface as diagnosis results for the user to see. The user interface comprises of five (5) major components which includes: (i) the problem input field that allows the user to enter the problem to be submitted to the system and it is editable i.e. allow the user to modify his problem, (ii) the Diagnosis button that normally starts the diagnosis operation by sending the problem to the system when fired, (iii) the diagnosis results area is a portion of the user interface that contains the results of the problem being retrieved based on their relevancy to the user given problem in a ranked order, (iv) the diagnosis result content portion that displays the content of the diagnosis result by simply selecting any of the problems in a diagnosis results portion of the user interface, and (v) the similarity field designed to display the degree of similarity value of each solution that matched the problem submitted by the user.

Problem Preprocessing: Document or text indexing is the process of transforming a document text into a representation of text and it may involves three major steps of tokenization, stop word filtration and stemming (Garcia, 2005). The problem text need to be preprocessed to extract the terms needed to be used as index for enhancing the process of matching the problem terms with relevant cases in a system case base. To achieve such objective, there are three steps (i.e. tokenization, stop words filtration and stemming) that have to be accomplished.

Tokenization: In the tokenization step, the user entered problem text is to be broken into tokens of lowercase letters. This enables the removing of any punctuation symbols, numbers or even hyphens in the problem. These tokens are to be passed to the next step of stop words filtration for further processing.

Stop Words Filtration: The tokens produced by the tokenization step consist of a number of frequent terms in the problem referred to as stop-words. These stop words are to be filtered out at this step since they might not contribute in identifying the content of the cases during retrieval process. For instance, the problem "my keypad is not working" consists of "my", "is", and "not" as stop words that has to be removed and simply become as "keypad and working".

Stemming: The stemming step involves reducing the problem tokens or terms that are not stop words into their roots. For instance, the terms "worker", "working", "worked" and "works" are to be reduced to "work" during this step. This will give a better matching of the problem terms with the cases to be retrieved during the retrieval process.

Search Engine: The search engine used for the diagnosis support system to provide the index and search capabilities to the system is the Lucene search engine. According to Recio-Garcia *et al* (2008) the Apache Lucene search engine is based on the statistical similarity approach that yields good results in the Information Retrieval (IR) field. It used both the vector space model and Boolean model (Manning *et al*., 2009) of Information Retrieval during the process of determining how relevant a case to be retrieved is to the user given problem. Using Lucene becomes necessary because of the importance of good indexing contribution to the retrieval of relevant solutions during the process of diagnosing a given problem. The search engine gets its input (problem terms) from the preprocessing stage and then used the *luceneTextualsimilarity* package (Source Forge, 2009) that is based on nearest neighbor (Beyer *et al*., 1999; Athitsos *et al*., 2008) retrieval approach to compare the problem to the set of cases that are already indexed by the lucene with a view to computing their similarities with the problem. Having gotten the similarity of the problem to each case, then the *NNretrieval.NNScoring* package (Source Forge, 2009) would be used to retrieve the topmost relevant solutions (i.e. those with highest similarity values) to the problems as the diagnosis results. These results are sent by the search engine to the user interface as a list of problems and solutions ranked based on their relevance to the user entered problem.

Case-Base: The diagnosis support system case base is a collection of previous cases of mobile phone problems and their solutions that the system used to base its reasoning while diagnosing a new problem. The case base is stored in a single text file consisting of cases arranged as problem and solution pairs as shown in Figure 2, where by each case in the text file is being mapped to its equivalent attribute in the application Case Bean Class. A customized connector is used by the application to read the entire case base into the memory for it to be accessible by the lucene search engine. The Lucene created inverted index terms with a view to indexing all the system cases before comparing them with the user given problem in order to retrieve relevant solutions and present them to the users via the interface.

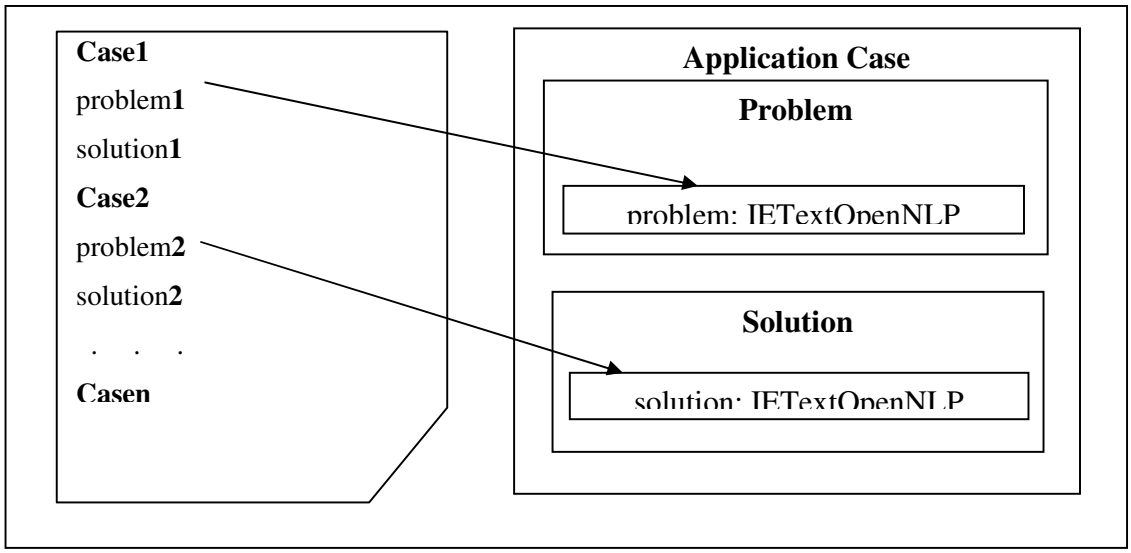

Figure 2: Case Structure

System Implementation: This section describes the actual implementation of the mobile phone diagnosis support system components based on the detailed design specified in Figure 1. The system is implemented using the jCOLIBRI CBR framework (Source Forge, 2009). The source codes of some classes in jColibri framework were modified and integrated within the new system to suit the purpose of the proposed system. The system components are discussed and specifically how they are implemented to serve the needed functionalities of the developed system.

Diagnosis Support System Interface: The Diagnosis Support System Interface was implemented using Java Language and integrated with other system components by reusing and extending jCOLIBRI framework. Figure 3 shows an example of the process of diagnosing a given problem. The user can first enter a problem description and after submitting the problem, a collection of cases which match the solution description are retrieved and returned to the user. These cases are ranked based on their similarities to the problem submitted, so the user can select any similar problem on the interface to see if it is the corresponding solution.

Problem Preprocessing Components: The preprocessing operations on the problem text are performed using the textual packages provided by the jCOLIBRI Framework. These packages consist of classes and their various methods to ease this operation. The *OpennlpSplitter* package (Source Forge, 2009) receives the problem and organized it into a stream of tokens by eliminating any punctuation symbols, numbers and so on. This is known as the tokenization. The *StopWordDetector* package (Source Forge, 2009) removes the stop words in the problem tokens. Then the *TextStemmer* package (Source Forge, 2009) performs the stemming on those tokens that are not stop words. All these operations are carried out during this stage to make the problem easier to be match with the similar cases in the application case base.

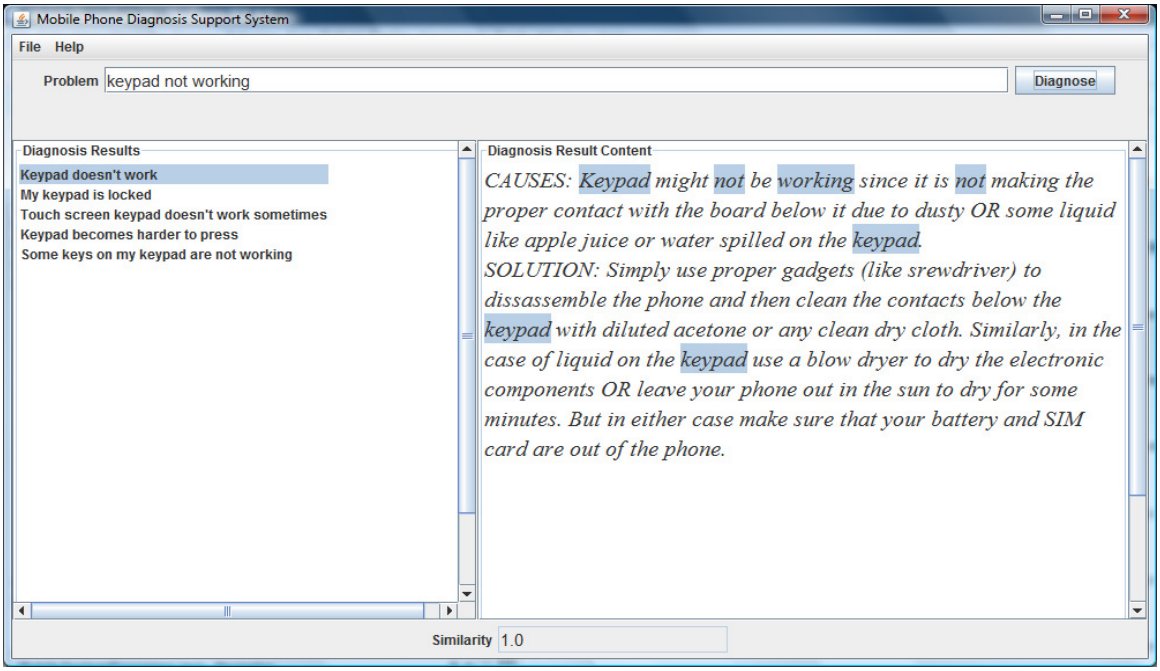

Figure 3: A Snapshot of Problem Submission in Diagnosis Support System

Search Engine Component: The search engine is implemented simply by reusing the lucene textual package provided in the framework. First of all the Connector in the *configure()* method of the application loads the textfile containing the cases via its specified path into the memory. This textfile is the case base used by the system. For the purpose of indexing these cases the lucene method need an index to be created using the *jcolibri.method.precycle.LuceneIndexCreator* method in the *precycle* method (Recio-Garcia *et al.*, 2008). Therefore, the luceneIndex variable is created of type *LuceneIndex* class to index all the cases in the System Case base as inverted terms index for easy comparison with the user given problem. Once a problem is received, the *LuceneTextSimilarity* package compares the problem with the already indexed cases to determine its closeness to the cases by computing their similarities. The *NNScoringMethod.evaluateSimilarity* method retrieved the solutions with the highest similarity values as the diagnosis results to be presented to the user.

System Case Base Component: In jCOLIBRI cases can be stored in text files, database files and even XML files format as case base. Here, the case base is implemented as a *LinealCaseBase* that stores the cases as a list data structure by reading the text file cases. The *PlainTextConnector* of the framework is used to manage the application cases stored in the text file as described in section 3.4.

System Evaluation: After implementing the developed diagnosis support system, performance of the system has been evaluated based on the degree of the similarity matching and the relevant of the textual solution retrieved to the given problem respectively. This evaluation is done using some collection of 42 mobile phone problem cases compiled from some phone repairers within Sokoto metropolis whom normally repairs the users' phone.

Evaluation Criteria: The essence of the evaluation is to assess the effectiveness of the system in retrieving the relevant solutions from its case base to diagnose the user given problem. Precision and recall measures were

used to assess the effectiveness of a system (Ghowdhury, 2004). In this case, the system is evaluated according to the following criteria:

Precision: The number of solutions retrieved that are relevant.

Recall: The number of relevant solutions that are retrieved.

Similarity Degree: The similarity matching value of the solution retrieved to the problem.

The experiment is carried out using six (6) domain experts (repairers) and six (6) users, by allowing each expert or user to submit one problem for the system to diagnose. So whenever each of the problems is submitted, the expert or user was asked to identify the relevant solutions being retrieved by the system to that problem. The performance of the system is captured by taking the precision, recall and similarity degree of the retrieved solutions into account.

RESULTS AND DISCUSSION

Results in Figure 4, 5 and 6 respectively have shown the precision, recall and similarity degree values obtained during the experiment for all the 12 problems submitted by the experts and users.

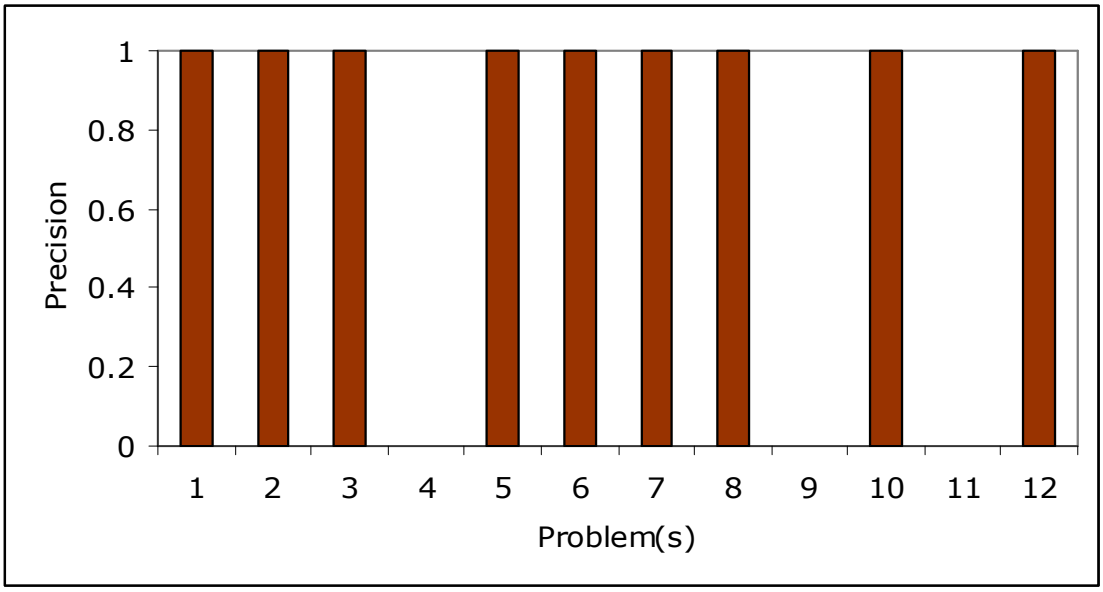

Figure 4: Problems Vs Precision Experimental Results

In Figure 4, the results have shown that 75% of the problems have high precision and 25% of the problems have low precision values. As we can see from the Figure, the system has shown high relevant solutions retrieval performance except with problem 4, 9 and 11 respectively where the system retrieves irrelevant cases. This implies that, most of the solutions that the system retrieved are relevant to the users given problems.

In Figure 5, the results have also shown that 75% of the problems have high recall and 25% of the problems have low recall values. As we can see from the Figure, the system has shown high solutions retrieval performance except with problem 4, 9 and 11 respectively where the system retrieves irrelevant cases. This might occur due to the limited collection of cases available in the case base during the experiment. This implies that, the system retrieves irrelevant solutions to answer the users given problems if there is no any solution that is relevant from its case base.

In Figure 6, the results have shown that 75% of the problems have high similarity with the retrieved solutions and 25% of the problems have low similarity values. As we can see from the Figure, the system has shown high

similarity relationships of the problems with the solutions retrieved except with problem 4, 9 and 11 respectively where the system shows low similarity relationship. This implies that, irrelevant solutions could never have high similarity since they are not relevant to the users given problems.

Therefore, it is expected that, the system will work far better when more cases are added in the future during the diagnosis of users given problems. On the whole, the results suggest that the developed system performs better in terms of precision, recall and similarity results obtained in this work.

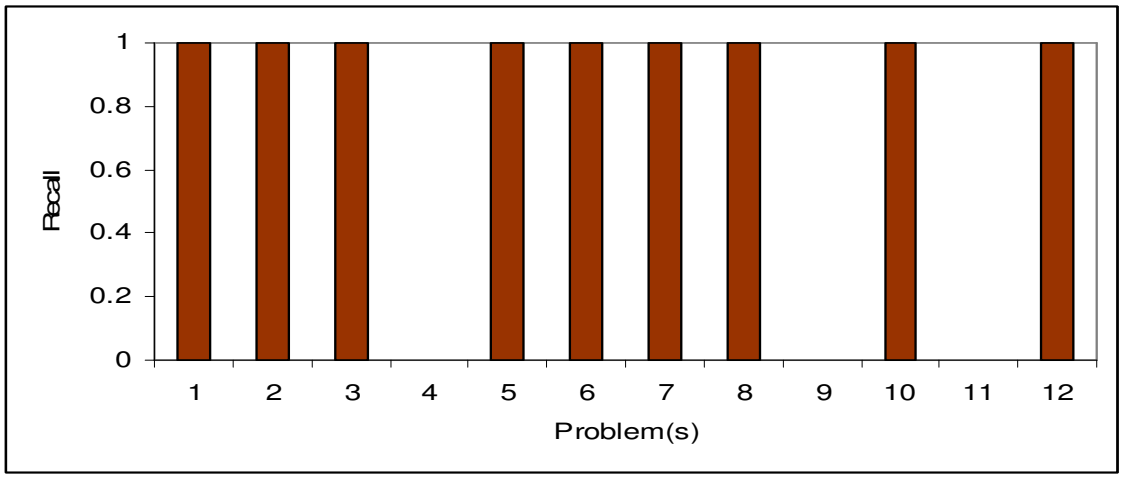

Figure 5: Problems Vs Recall Experimental Results

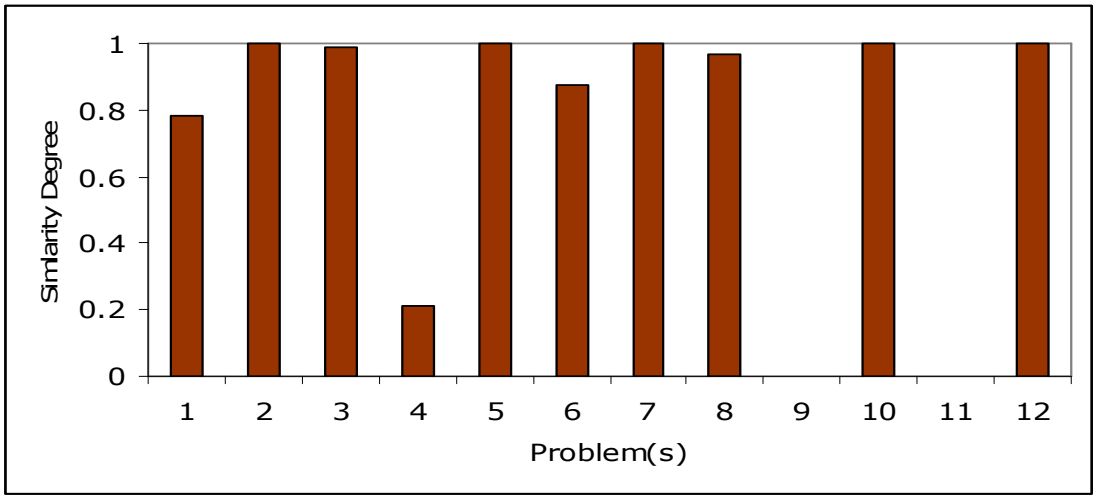

Figure 6: Problems Vs Similarity Degree Experimental Results

Conclusion: In this paper, a diagnosis support system is presented that used case based reasoning for the diagnosis of mobile phone problems. The system uses its preprocessing component to break a problem into a representation of text suitable for enhancing the process of matching the problems with the relevant cases in its case base by the search engine during the diagnosis of each problem. Experiments with a collection of some cases show that the system diagnosis a problem with a relevant solution based on the precision, recall and similarity results obtained.

The work provides some interesting insights into the task of developing a simple Textual CBR system, and the potential benefits of jCOLIBRI. It demonstrated the simplicity of jCOLIBRI in developing Textual CBR applications by reusing or extending existing packages. Similarly, the system is easily modifiable to suit any problem domain that works in form of question and answering format without writing even a single line of code by simply modifying or replacing the content of the case base.

However, as a future work, the research will focus on integrating knowledge source to the developed system in order to take the synonyms similarity of the problem terms into account while diagnosing a given problem, since the same problem could be expressed using different words.

REFERENCES

Athitsos, V., Alon, J., Sclaroff, S. and Kollios, G. (2008). BoostMap: *An Embedding Method for Efficient Nearest Neighbor Retrieval*. IEEE Computer Society. 30(1) pp.1-16.

Bergmann, R., Kolodner, J. and Plaza E. (2005). Representation in Case-Based Reasoning. *The Knowledge Engineering Review*. United Kingdom: Cambridge University Press. 00(0) pp. 1-4.

Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999). When Is "Nearest Neighbor" Meaningful? *Berlin: Springer-Heidelberg. pp. 217-235.*

Garcia, E. (2005). Document Indexing Tutorial: *Document Indexing Tutorial for Information Retrieval Students and Search Engine Marketers*. [online] Available from: http://www.miislita.com/information-retrieval-tutorial/indexing.html [Accessed May 24, 2009]

Ghowdhury, G. G. (2004). *Introduction to modern information retrieval*. 2nd ed. London: Facet publishing.

Lenz M. (1998). Textual CBR and Information Retrieval – A Comparison. In: *Proceedings of sixth German Workshop on Case-based Reasoning*. Berlin: Springer. Pp. 1-8

Lenz, M., Hubner, A. and Kunze, M. (1998). Question and Answering with Textual CBR. In: *Proceedings of the Third International Conference on Flexible Query Answering Systems.* Berlin: Springer-Verlag. pp. 236-247.

Manning, C. D., Raghavan, P. and Schutze, H. (2009). *An Introduction to information retrieval*. England: Cambridge University Press. [online] Available from: http://nlp.stanford.edu/IRbook/pdf/irbookonlinereading.pdf [Accessed May 25, 2009]

Pal, S. K. and Shiu, S. C. K. (2004). *Foundations of soft case-based reasoning*. Canada: John Wiley and Sons. Pp. 3-12

Recio-García, J. A., Díaz-Agudo, B. and González-Calero, P. (2008). jCOLIBRI2 Case Based Reasoning Framework. *jCOLIBRI2 Tutorial*. Group for Artificial Intelligence Applications, Universidad Complutense De Madrid. Available from: http://gaia.fdi.ucm.es/projects/jcolibri/jcolibri2/tutorial.pdf[Accessed June 23, 2009]

Source Forge. (2009). jCOLIBRI: *CBR Framework*. [online] Available from: http://sourceforge.net/projects/jcolibri-cbr/files/jCOLIBRICBR/jCOLIBRI21.zip/ [Accessed June 22, 2009]

Weber, R. O., Ashley, K. D. and Bruninghaus S. (2006). Textual Case-Based Reasoning. *The Knowledge Engineering Review.* United Kingdom: Cambridge University Press. 20(3) pp. 255-260.