International Journal of Computer Science Research and Application
2013, Vol. 03, Issue. 03, pp. 02-09
ISSN 2012-9564 (Print)
ISSN 2012-9572 (Online)

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE RESEARCH AND APPLICATION

www.ijcsra.org

Online Question Answering System

Bolanle Ojokoh[1], Peter Ayokunle[2]

[1]bolanleojokoh@yahoo.com
[2] peter.ayokunle@gmail.com
Department of Computer Science, Federal University of Technology, Akure, Nigeria.
bolanleojokoh@yahoo.com

Abstract

Owing to the vast amount of information readily available on the World Wide Web, there has been a significant increase in the number of online question answering (QA) systems. A branch of QA systems that has seen such remarkable growth is the community-based question answering (cQA) systems. In this paper, we propose a method that is proactive enough to provide answers to questions and additionally offers word definitions, with the aim of reducing the time lag that results from askers having to wait for answers to a question from various users. The evaluation results on the predicted answer from the computing-related datasets employed, demonstrate the effectiveness of the proposed technique.

***Keywords**:* Question answering, Community-based question answering, Askers, Answerers, Voters.

1. Introduction

Question answering is a computer science discipline within the fields of information retrieval and natural language processing. It is the task of automatically answering a question posed in natural language. QA systems deliver users short, succinct answers instead of overloading them with a large number of irrelevant documents [1]. This is the goal of every QA system. More commonly, QA systems pull answers from unstructured collection of natural language document. An interesting branch of these systems which has recently evolved is the cQA systems from which the datasets used in this work were extracted. cQA services are dedicated platforms for users to respond to other users' questions, resulting in the building of a community where users share and interactively give ratings to questions and answers[2]. From observation, more users are becoming inclined to this new area of QA systems as they can obtain a near perfect answer to their questions from other users rather than a list of likely document containing result(s) to their question that are provided by the system. While current web search engines enjoy huge commercial success and demonstrate good performance, especially for homepage-finding queries, their ability to find relevant information for hard queries such as those asking for opinions or summaries is far from satisfactory [3]. These complicated user information needs can be satisfied by using QA services.

Many of the existing works on QA systems has focused on retrieving high quality answers to an asker's question and recommending answerers [4]. However, the time lag that results from askers having to wait for answers to a question from various users as identified by [3] have not received much consideration. In addition, while browsing the internet, we often spend large amount of time searching for a particular answer that best answers our question, and end up getting too large materials that are not specific and so difficult to extract specific and quality answers to the questions we have. This paper focuses on reducing this time lag to a bare minimum by selecting the most relevant question and answer pair (from previously answered questions) to an asker's query before other users are available to provide answers to such query. It uses text similarity measures (Dice metric and a modified form of Levenshtein distance) for relevant answer retrieval. It additionally offers definitions of words and their synonyms when the "define" keyword precedes the query using the WordNet database. Consequently, this additional feature brings dictionary functionality into proposed the QA system. Section two reviews related works. Section three describes the architecture of the proposed system. Section four contains the experiments and evaluation while section five concludes the work and proposes areas of further research.

2. Related Work

Question Answering Systems have been in existence for several decades now [5], [6],[7]. [8] developed the first question answering system (called START, SynTactic Analysis using Reversible Transformations) for the World Wide Web.START has been in continuous operation since December, 1993 and has answered millions of questions from hundreds of thousands of users all over the world. There has been extensive research on question answering(QA) since then [9][10].The traditional QA solutions such as in [11] are often times content-based and focus on searching and extracting answers from a static collection of text segmentsor web data [12]. Some QA works are ontology driven. For instance, [13] developed an ontology driven question answering system called AQUA (Automated QUestion Answering system). AQUA uses NLP technology, logic and a hand-crafted ontology to find a textual answer to the question in a short period of time. AQUA uses the semantic annotations to perform inferences and reduce the number of possible answers to a question. The major contribution of AQUA is the use of ontology in order to go beyond superficial keyword matching as typical search engines.

A number of works has been undertaken by researchers in recent years on QA systems. A large chunk of these works as mentioned earlier has been geared towards methods for recommending high quality answers [2], recommending answerers and identifying research issues with QA systems [14]. Since this work adopts text similarity measures in providing the most related question and answer pair, we therefore provide related works in this regard. In [15] proposed a method that allows taking advantage of the output of several QA systems. This method is based on an answer validation approach that decides about the correctness of answers based on their entailment with a support text, and therefore, that reduces the influence of the answer redundancies and the system confidences. In order to reduce the support text to the minimum useful text fragment, they captured the morphological inflections of words using Levenshtein edition distance. They considered that two different words are equal if its distance value is greater than 0.6. In [16] presented an approach which, given a knowledge base and an appropriate text corpus, automatically induces patterns which can be used to query the knowledge base. Given a partial frame and a matching predicate-argument structure, they determined the highest-scoring mapping between attributes of the partial frame and arguments of the predicate-argument structure by calculating the average Levenshtein distance. In [17] estimated the impact of dealing with automatically transcribed (i.e. noisy) requests on a specific question interpretation task, namely the extraction of relations from natural language questions. They employed Levenshtein distance in calculating the probability of an entailment relation between question q and a pattern p which is related to the possibility of mapping the whole content of q into the content of p. The more straightforward the mapping can be established, the more probable is the entailment relation. Moreover in [18] proposed an approach which uses simple pattern matching rules for the answer extraction phase of a passage-based search engine. Some spell-checking function was added in this phase (answer extraction) by using Levenshtein distance to compare strings before passing the string to a text crawler. The tools used in this work for comparing and selecting the most relevant question and answer pair are however not limited to Levenshtein distance alone but also involves the Dice metric and a parts-of-speech (POS) tagger.

3. The Proposed System

Figure 1 gives the architectural overview of the proposed system. Each of the segments of the Question Answering architecture is described in the sub sections that follow:

3.1 Question Processing

The question processing module is initiated when a user enters a query (question) in the provided search box. The format of the question is as we have in any cQA system in that there are no strict rules to how a question can be asked. For example, a user who has a challenge turning on his playbook device may go ahead and ask a question as “Playbook trouble! Can’t seem to power up my device, any help please???????” The user completes the question processing module by pressing the return key or the “Ask” button provided by the system which consequently alerts the server of an incoming request.

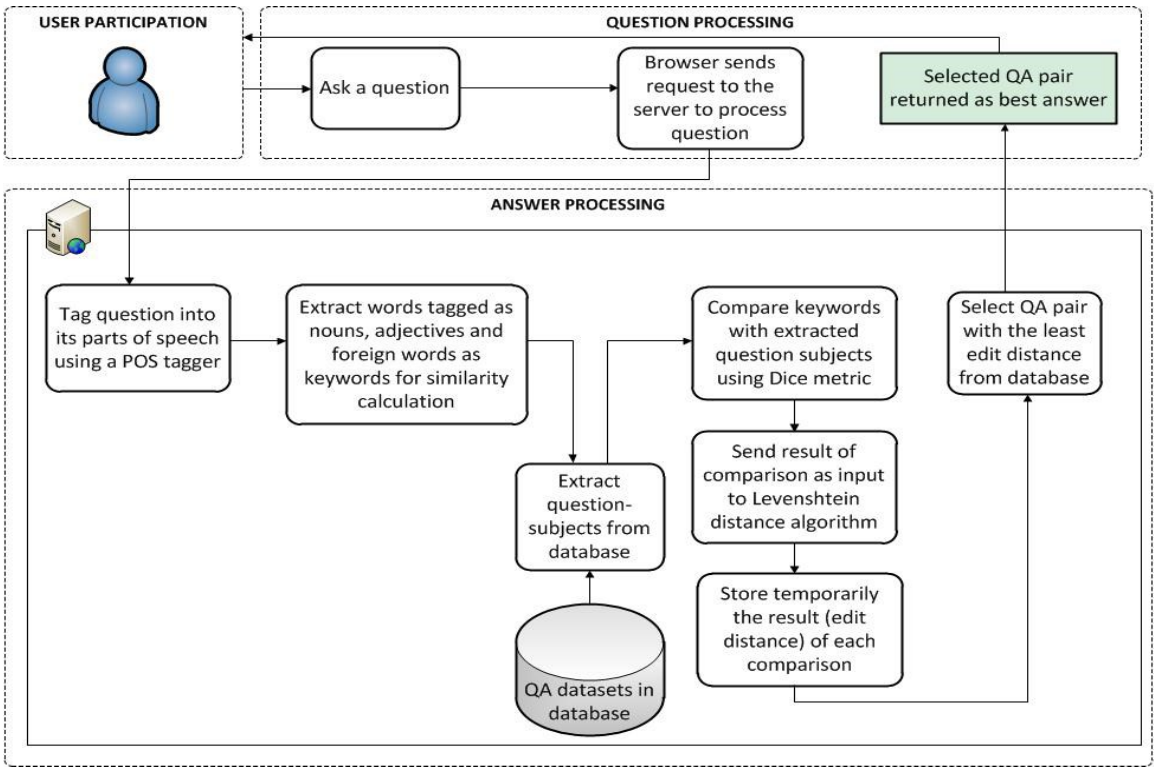
USER PARTICIPATION
QUESTION PROCESSING
Ask a question
Browser sends request to the server to process question
Selected QA pair returned as best answer
ANSWER PROCESSING
Tag question into its parts of speech using a POS tagger
Extract words tagged as nouns, adjectives and foreign words as keywords for similarity calculation
Extract question-subjects from database
QA datasets in database
Compare keywords with extracted question subjects using Dice metric
Send result of comparison as input to Levenshtein distance algorithm
Store temporarily the result (edit distance) of each comparison
Select QA pair with the least edit distance from database

Figure 1: Proposed system architecture

3.2 Answer Processing

In the answer processing phase, the query is first tagged into its parts of speech using the OpenNLP Library POS tagger. A POS tagger is a piece of software that reads text in some language and assigns parts of speech to each word (such as noun, verb, and adjective) of a sentence. From observation, keywords in a query in cQA systems are expressed as nouns, proper nouns, adjectives and foreign words. As a result, words corresponding to these parts of speech in the tagged question are extracted as keywords which will be used in calculating the similarity between the question and the available question and answer pair in the database. The POS tagger used in this work uses the same tag abbreviations as used in the Penn Treebank Project [9]. The tags and their corresponding descriptions are shown in Table 1.

Table 1: The POS tagger coded abbreviations

Tag	Tag Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item maker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural

PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	To
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Question-subjects are extracted from the available datasets in the database. As each question-subject is extracted, it is compared with the keyword(s) using the dice metric. The result (output) obtained from here is further passed as input to the modified Levenshtein distance algorithm. The Levenshtein distance algorithm was modified in this work because in its original form, the algorithm does a character-character comparison of strings and since the proposed system works with keywords rather than characters; there arose the need to tweak the algorithm accordingly. The result (edit distance) of each comparison is then stored temporarily (e.g. in an array) alongside the extracted question-subjects. The question-subjects are stored temporarily in order to serve as key for each edit distance calculated. This will enhance seamless retrieval of the best QA pair from the database later. Once all question-subjects have been extracted, compared and temporarily stored, the minimum of these edit distances is calculated. This is done to identify the question-subject that is the shortest and most similar to the query. The QA pair matching this minimum edit distance is then selected and returned to the user using the temporarily stored question-subjects as the key for retrieval. If the answer processing module spots the "define" keyword at the beginning of the query, it redirects to the WordNet database [20] to retrieve the definition of the word in question and its synonym and the result is returned together with the matching QA pair earlier extracted to the happy user.

3.3 Dice algorithm

1. Get the two strings (say A and B) whose similarity is to be determined.
2. Extract keywords from string A.
3. Detect the number of keywords of A that are in B (i.e. $A \cap B$)
4. Multiply the result from step 3 by 2
5. Determine the length of string A and B by counting the number of words in each.
6. Add the length of A and B together.
7. Divide the result from step 4 by the result from step 6.

3.4 Modified Levenshtein distance algorithm

1. Get the two strings (string1, string2) to compare.
2. Split the two strings using a whitespace as delimiter (i.e. tokenize the two strings).
3. Remove any trailing "?", "!", "." from the last word in both strings.
4. Store the tokens from both strings separately (i.e. tokensInString1, tokensInString2).
5. Loop through all elements in tokensInString1.

6. For each iteration in tokensInString1, test if a word (i.e. an element) is contained in tokensInString2. Also, test if the word is not a stopword (words which are filtered out prior to, or after, processing of natural language data (text))
   If yes, increment a counter (initially set to zero)
   Else do nothing and proceed to the next iteration
7. At the end of the iteration, return counter.

3.5 User Participation

It is evident from the question processing and answer processing modules that user participation forms the backbone of CQA services (Blooma &Kurian, 2011).Without the interaction of users, the cQA system cannot realize its objectives. The datasets used in this work comprises users that participated by asking questions (askers), answering questions (answerers) and rating questions (voters) which aids the success of cQA services.

4. Experiments and Evaluation.

4.1 Data and Tools

This work uses as its dataset, 2000 question and answers (QA) pairs from Yahoo! Answers between the years 2009 and 2010, downloaded from sourceforge [21]. The QA pairs consist of questions and answers from different categories of Computing. The characteristics of the QA pairs are further described in Table 2.

Table 1: Characteristics of the Dataset

Hardware	**No. of Questions**	**Internet**	**No. of Questions**
Add-ons	277	FaceBook	205
Laptops & notebooks	384	Google	213
Printers	211	Wikipedia	201
New Category	311	You Tube	198
Total	1,183		817
Overall = 2,000 dataset			

4.2 Experiments

Experiments were carried out using question subjects from various categories (Add-on, facebook, google, laptops & notebooks, new category, printers, wikipedia, and youtube) into which the dataset was classified. The question subjects from each category were supplied as user's query. The experiment was also repeated using the unmodified Levenshtein distance algorithm only (i.e. the algorithm in its original form) as well as with the modified version of the algorithm. The results were then compared with that of our system.

4.3 Evaluation

In the evaluation of the answers returned, a java code was written to extract question subjects from the 2000 datasets used and fed back to our system to check how well the system returns the same question subject and its corresponding answers as its response. The higher the semantic relationship between these question subjects and responses, the better the system performance. To test the dictionary functionality of the system, the define keyword was manually entered before certain common words (e.g. laptop, internet etc) and the result was verified. This evaluation process was repeated for the two other approaches (unmodified and modified Levenshtein distance algorithm) with which the results were compared. The precision (P), recall (R), f-measure (F) (commonly used in the information retrieval community [22]) and Mean Reciprocal Rank (MRR) were calculated using the formula below.

$$\text{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{retrieved}|\text{relevant}) \quad (1)$$

$$\text{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant}|\text{retrieved}) \quad (2)$$

$$F = \frac{2PR}{P+R} \quad (3)$$

MRR is a statistical measure for evaluating any process that produces a list of possible responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q. The reciprocal value of the mean reciprocal rank corresponds to the harmonic mean of the ranks.

MRR is expressed mathematically as:

$$MRR = 1/\#Q \left(\sum_{q \in Q} \text{rank}_q^{-1}\right) \quad (4)$$

where rank_q is the first rank at which correct answers occur for q in query question set #Q. These results are shown in Table 3 and Figures 2a and 2b.

Table 3: Comparison of results returned using some standard evaluation measures

Method	Categories	Recall	Precision	F-measure	MRR	No. of questions
Unmodified Levenshtein distance	Add-on	0.004	1.000	0.008	0.004	277
	Laptops&Notebooks	0.000	0.000	0.000	0.000	384
	Printers	0.000	0.000	0.000	0.000	211
	Facebook	0.000	0.000	0.000	0.000	205
	Google	0.000	0.000	0.000	0.000	213
	Wikipedia	0.000	0.000	0.000	0.000	201
	Youtube	0.000	0.000	0.000	0.000	198
	New category	0.000	0.000	0.000	0.000	311
	Average	0.001	0.125	0.001	0.001	
Modified Levenshtein distance only	Add-on	0.881	0.996	0.935	0.876	277
	Laptops&Notebooks	0.859	0.997	0.923	0.858	384
	Printers	0.844	1.000	0.915	0.839	211
	Facebook	0.883	0.989	0.934	0.884	205
	Google	0.859	0.995	0.922	0.858	213
	Wikipedia	0.891	0.984	0.935	0.891	201
	Youtube	0.919	0.995	0.955	0.921	198
	New category	0.910	0.993	0.950	0.912	311
	Average	0.881	0.994	0.934	0.880	
(Proposed Method) Dice + Levenshtein distance + POS tagging	Add-on	0.881	0.992	0.933	0.880	277
	Laptops&Notebooks	0.875	0.997	0.932	0.875	384
	Printers	0.882	0.995	0.935	0.878	211
	Facebook	0.888	0.989	0.936	0.886	205
	Google	0.892	0.990	0.938	0.895	213
	Wikipedia	0.980	0.995	0.987	0.982	201
	Youtube	0.939	1.000	0.969	0.937	198
	New category	0.961	0.997	0.979	0.962	311
	Average	0.912	0.994	0.951	0.912	

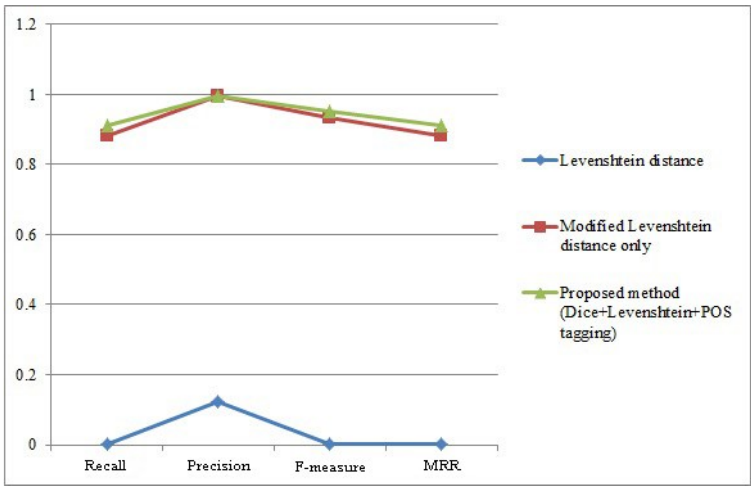

Figure 2a: A comparison of the different methods employed

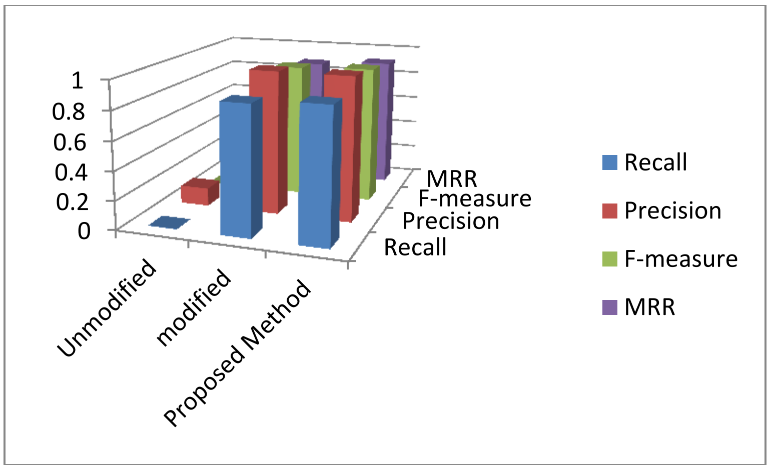

Figure 2b: A comparison of the different methods employed

For this work, the following rule was used:

Score for individual question i is the reciprocal rank ri where the first correct answer appeared (0 if there is no correct answer in top 3 returns).

The repeated list of zeros and mean reciprocal rank of 0.1% in Table 3 further proves that in its raw form, the Levenshtein distance algorithm can not be applied to this work without modification since any slight change to just a character in the user query will greatly reduce the possibility of matching the most relevant question (since Levenshtein distance is efficient for character-character comparison of words). Also, a high recall rate for the combined algorithm (i.e. the proposed system) shows that about 91% of the time, the developed system retrieved more relevant documents than none. Similarly, a mean precision of 0.994 suggests that about 99% of the time, the system returned a high number of relevant documents in the retrieved document. A mean reciprocal value of 0.912 shows that the system answered 91% of the questions correctly. This is a good performance for the system.

Although, aside Levenshtein distance, dice metric or other similarity algorithms, some of the existing approaches of determining high qualities answers to a question in a cQA uses some other methods like markov chains, who-answers-who relationship, user-prestige among others. However, our proposed method can still be compared with some related systems. It performs better in both precision and recall (0.994, 0.912) as compared to 0.794 and 0.771 for high quality questions recorded in [23]. Also, our system records a better performance in terms of precision, recall and f-measure (0.994, 0.912, 0.951) when compared to [24] which gave (0.717, 0.879, 0.790) respectively. In TREC-8, 9, 10 [1], best systems returned MRR of 0.65-0.70; an MRR of 0.912 shows that our system performs better since the closer this value is to 1, the better the performance of the system.

5. Conclusion

In this paper, we proposed an approach for retrieving high-quality answers using dice metric and a modified form of Levenshtein distance algorithms. In addition, our technique offers word definitions aimed at reducing the time lag earlier identified. The evaluation results from experiments indicate that the proposed techniques are effective at achieving this goal. In the future, we intend to extend the scope of the system to cover questions from other areas of life such as arts, entertainments, animals, food and drink, games, health etc.

References

1. Lin, J., Katz, B.: Question answering techniques for the World Wide Web. In: proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics, (2003)
2. Wang, X., Zhang, M.: Let other users help you find answers: A collaborative question-answering method with continuous markov chain model.In: proceedings of the Asia-Pacific Web Conference, pp. 264-270, (2011)
3. Jeon, J., Croft, W. B., Lee, J. H. : Finding similar questions in large question and answer archives. In: proceedings of the 14th ACM Conference on Information and Knowledge Management, Bremen, Germany, pp. 84-90.ACM, NY (2005)
4. Hieber, F., Riezler, S.: Improved Answer Ranking in Social Question-Answering Portals. In: Proceedings of SMUC '11, October 28, 2011, Glasgow, Scotland, UK (2011)
5. Green, W., Chomsky, C., Laugherty, K.: BASEBALL: An automatic question answerer. Proceedings of the Western Joint Computer Conference, pp. 219-224 (1961)
6. Woods, W. A. :Progress in natural language understanding: an application to lunargeology. In: Proceedings of the June 4-8, 1973, national computer conference and exposition,AFIPS '73, New York, NY, USA, New York:ACM, pp. 441–450. ACM, (1973)
7. Wilensky, R., Ngi-Chin,D., Luria, M., Martin, J.H., Mayfield, J. Wu, D.: TheBerkeley UNIX Consultant project. Technical Report CSD-89-520, Computer Science Division, the University of California at Berkeley, (1989)
8. Katz, B., Lin, J., Felshin, S: Gathering Knowledge for a Question Answering System from Heterogeneous Information Sources. In proceeding of the ACL 2001 workshop on human language technology and knowledge management, (2002)
9. Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R., Harabagiu, S., Israel, D., Jacquemin,C., Lin,C.Y., Maiorano,S., Miller, G., Moldovan, D., Ogden, B., Prager ,J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E., Weishedel, R.:Issues, Tasks and Program Structures to Roadmap Research in Question Answering (QA), (2002)
10. Dang N.T., Tuyen, D.T.T.: Natural Language Question Answering Model Applied To Document Retrieval System, (2009)
11. Agichten, E., Lawrence,S., Gravano, L.:Learning to Find Answers to Questions on the Web. ACM Transactions on Internet Technology, (2002)
12. Suryanto, M. A., Lim,E. P., Sun, A., Chiang, R. H. L.: Quality-Aware collaborative question answering: methods and evaluation. In: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in InformationRetrieval, Barcelona, Spain, New York:ACM, pp. 142-151, (2009)
13. Vargas-Vera, M., Motta, E., Domingue, J.: AQUA: An Ontology-Driven Question Answering, (2003)
14. Blooma, M. J., Kurian, J. C.:Research issues in community based question answering. In proceedings of the Pacific Asia Conference on Information Systems , pp. 29, (2011).
15. Valero, A., Montes-y-Gómez, M., Villaseñor-Pineda L., Anselmo P.:. Towards multi-stream question answering using answer validation. Journal: Informatica (slovenia) - INFORMATICASI, vol. 34, no. 1, (pp 45-54), (2010).
16. Cimiano, P., Erdmann, M., Ladwig, G.: Corpus-based Pattern induction for a knowledge-based question answering approach. In proceedings of the 1st IEEE International Conference on Semantic Computing (ICSC) , pp. 671-678, (2007).
17. Gretter, R., Kouylekov, M., Negri, M.: Dealing with spoken requests in a multimodal question answering system. In proceedings of the 13thInternational Conference AIMSA pp. 93-102, (2008).
18. Soriano, J. M. G., Buscaldi, D., Asensi, E. B., Rosso, P., Arnal, E.S.: QUASAR: The question answering system of the Universidad Politecnica de Valencia. In proceedings of the Sixth Workshop of the Cross-Language Evaluation Forum, pp. 439-448, (2005).
19. PennTreeBankProject, http://ling.upenn.edu/courses/Fall_2003/ling001/penn
20. WordNet database, http://wordnet.princeton.edu/
21. Sourceforge, Open Source Applications and Software Directory, http://sourceforge.net
22. Ojokoh, B., Zhang, M., Tang, J. A Trigram Hidden Markov Model for Metadata Extraction from Heterogeneous References. Information Sciences, vol. 181, pp. 1538 – 1551, (2011).
23. Agichtein, E., Castillo, C., Donato, D., Gionis, A., Mishne, G.: Finding high-quality content in social media. In proceedings of the International Conference on Web search and web data mining, pp. 183-194, (2008)
24. Shrestha, P. : Corpus-Based methods for Short Text Similarity. Researchers students meet computer for automatic processing of language, pp. 297, (2011)