COMPARISON OF STRING SIMILARITY ALGORITHMS TO MEASURE LEXICAL SIMILARITY

Mr. Sagar J. Gandhi, Mr. Mihirraj M. Thakor, Dr.Jikitsha Sheth, Mr, Hariom I. Pandit, Mr. Hemin S. Patel

Abstract—A string similarity represents the lexical similarity between two words. This can be further exploited to identify similarity between questions. Several string similarity algorithm exists in literature. In this paper the authors have implemented five string similarity algorithms viz. Dice coefficient, Jaccard similarity, Levenshtein distance, Jaro distance and Cosine similarity. The results of these algorithms are further compared with human judges to determine, which of them resembles the human way to dissimilarize the given strings. The experimentation is done over 1000 English word pairs.

I. INTRODUCTION

Question Answering (QA) is a computer science discipline within the fields of Information Retrieval (IR) and Natural Language Processing (NLP), which is concerned with building systems that automatically answer the questions posed by humans in a natural language. Question Answering is in itself an intersection of Natural Language Processing, Information Retrieval, Machine Learning, Knowledge Representation, Inference and Semantic Search. A QA implementation, usually a computer program, may construct its answers by querying a structured database of knowledge or information, usually knowledge base. More commonly, QA systems can pull answers from an unstructured collection of natural language documents. QA research attempts to deal with a wide range of question types including: fact, list, definition, How, Why, hypothetical, semantically constrained, and cross-lingual questions [21].

A. Modules of QA:

QA domain deals to find answers for open-domain natural language questions in a large document collection. A typical QA system usually consists of three basic modules as shown in Figure 1: 1. Question Processing (QP) Module, which finds some useful information from questions, such as expected answer type and key words; 2. Document Processing (IR) Module, which searches a document collection to retrieve a set of relevant sentences using the key words; 3. Answer Extraction (AE) Module, which analyzes the relevant sentences using the information provided by the QP module and identify the proper answer. These three modules are shown in figure1 [30].

1) Question Analysis module

In question analysis module following tasks are performed:

- o Identify users question type
- o Determine answer type of users question
- o Identify topic and Theme from users question
- o Convert users question into IR query

o Identify users question type: The questions are usually categorized based on answer type, which means what the user needs in an answer, i.e. time, person, location, definition, numerical data, etc. Identification of question type helps other modules to get correctly locate the answer. Various techniques have been proposed by researchers for question classification such as Factoid, List, Definition, Relationship, etc. Factoids are the questions whose answer belongs to fact for person, object, place, numerical data, etc. For example, "Where is Eiffel Tower?", "What is the value of Pi?". Lists are questions whose answer returns the list of values. For example, "List of States in India." Definition is question whose answer returns the conceptual definitions and mostly contains question target. For example, "What is an Object?" Relationship is questions which returns the relationship type between the two objects. For example, "What is relationship between Ice and Water?" By this way, we can get which type of questions are asked by the user and also determine that which type of answer should be returned.

o Determine answer type of users question: For work of further modules, we must determine the answer type of user's question. The expected answer type can be defined based on question terms and rules such as

Who = Person; Where = Location, Place; How many = Number;

When = Time/Date; How=Method/Definition; Why= Explanation.

By the above given rules, identification of the answer type and process as per these determinations can be done.

o To identify topic and theme of the query : In this phase the main motive of the question identified. The module identifies the context of the question and accordingly the main theme of the question is recognized.

o Convert users question into IR query: This is the last step of question analysis, where the users' natural language query is converted into IR query, which is used to retrieve important documents.

2) Document Processing module

The quality of a selected paragraph is one of the critical parameter in question answering system. If the quality of paragraphs is poor, then the system returns to the question keyword extraction module, and alters the heuristics for extracting keywords from the question. Then the IR can be performed by using new set of key word retrieved from scratch. New queries for the information retrieval system are produced by revisiting the question keywords component, and either adding or dropping keywords. This feedback loop offers some form of retrieval context that ensures that only a 'reasonable' number of paragraphs are passed onto the Answer Processing module. Like several other parameters, exactly how many paragraphs constitute a 'reasonable' number should be configured, based on performance testing. Next, the paragraph ordering is done to rank the paragraphs according to a plausibility degree of containing the correct answer [30].

3) Answer Processing module

To identify useful information from the document, data extraction is done. Further, this module represents the data in more meaning manner and creates the answer that is to be shown to the user. From the number of answers created in such a manner, the one which ranks higher is shown first to the user. The ranking is purely based on relevancy [30].

B. Role of string similarity in QA :

Users generally give their question to the system in form a query. This query is given in natural language form rather than as a list of keywords. There are several questions available on website with appropriate answer. It is possible that some of them might be similar to the new query provided by the user. This results to question duplication on same the website. With this consideration, the present work focus on finding similarity between given two questions. To find question similarity, string similarity between given two questions has to be identified. In computer science, string similarity is a measurement of finding similarity distance between two strings using different mathematical string similarity algorithms.

Many times similarity is identified between strings that helps to reduce database and application load in real world systems. These mathematical string similarity algorithms give result in float numbers between 0 and 1. Some of these algorithms give result in the distance while some in form of similarity. String similarity is a main and primary stage for finding sentence similarity, paragraph similarity, etc. String similarity is used in many real life applications such as question-answering systems, fraud detection, plagiarism detection, OCR image analysis, data mining,

data integration, etc. String similarity can be measured with two contexts: Lexical and Semantic. Lexical similarity means similarity based on occurrences of characters in both the strings. While semantic similarity describes similarity between two strings based on their meanings. The next section discusses the string similarity algorithms based on lexical features of the string.

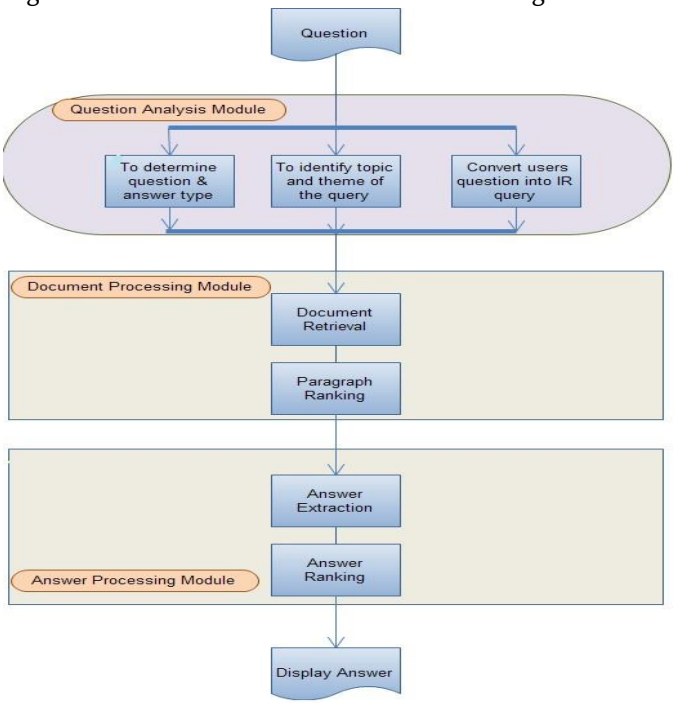


Figure 1. Architecture of QA system

II. ALGORITHMS TO FIND WORD SIMILARITY

1) Dice coefficient:

Dice coefficient is used for comparing the similarity of two strings. It was independently developed by the botanists Thorvald SØrensen and Lee Raymond Dice, who published in 1948 and 1945 respectively. The SØrense-Dice is also known as F1 score or Dice similarity coefficient (DSC).

Formula:

$$QS = \frac{2|X \cap Y|}{|X| + |Y|}$$

Using it, the average relevancy of each set of document for a single query is calculated in [26]]. It was found that cosine similarity gives better result that dice co-efficient.

2) Levenshtein distance:

Levenshtein distance algorithm is named after Vladimir Levenshtein, who considered this distance in 1965[27]. Mathematically, the Levenshtein distance between two strings a , b (of length |a| and |b| respectively) is given by leva,b (|a|,|b|) where (ai = bj) is the indicator function equal to 0 when ai = bj and equal to 1 otherwise, and leva,b (i,j) is the distance between the first i characters of string a and the first j characters of string b. In information theory and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

Formula:

Mathematically, the Levenshtein distance between two strings a, b is given by $\operatorname{lev}_{a,b}(|a|,|b|)$ where

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where $1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise.

It is the minimum cost of edit operations needed to transform one string into another [24].

3) Jaccard similarity:

The Jaccard similarity is a common index for binary variables. It is defined as the quotient between the intersection and the union of the pair wise compared variables among two objects.

Formula:

jaccard(A,B) = | intersection(A,B) | / | union(A,B) |

where A and B are strings. It can also be expressed in terms of true positives (TP), false positives (FP) and false negatives (FN) as:

jaccard(A,B) = TP / (TP + FP + FN)

4) Jaro distance:

Jaro distance algorithm proposed in 1999 by William E. Winkler of the Jaro distance metric (1989, Matthew A.Jaro) [28]. The higher the Jaro distance for two strings is, the more similar the strings are. The score is normalized such that 0 equates to no similarity and 1 is an exact match. The Jaro distance is a measure of similarity between two strings. m is the number of matching characters and t is half the number of transpositions.

Formula:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

5) Cosine similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0 is 1, and it is less than 1 for any other angle [29].

Formula:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

If there is more similarity between two documents, the cosine similarity value is near to 1 otherwise it is near to 0 [23].

III PROPOSED WORK

In this study, we present five algorithms for lexical word similarity algorithms (Dice coefficient, Levenshtein distance, Jaccard similarity, Jaro distance, Cosine similarity).The main objective of this paper is to find the best lexical word similarity algorithms. The system proposed model is shown in figure 2.

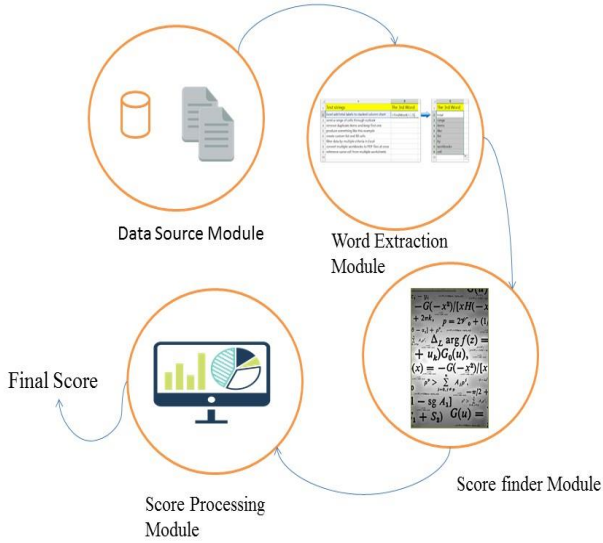


Figure 2. Proposed Model

A) Data Source Module:-

This module collects and stores the information from an external source and pass that information into the data stream. There are many types of data format available for the data storage like tables, tab-delimited file, worksheet file, CSV format file, etc. This module helps to maintain such heterogeneity of format in data. Hence, this module deals with storing of question pairs in different format as stated above.

B) Word Extraction Module:-

Once we have the data source ready with data, this module extracts the words from given question pairs and creates word pair. The word extractor will extract the pair of word and save them to data source if required by the user. Hence the output of the module will those words of pair which are stored into the data source or directly made available to the user.

C) Score finder module:-

After the word pairs are extracted, they are further sent to the score finder module. In this module, the similarity between given two words is found using all the similarity techniques discussed earlier.

Steps for score finder module:

1) Take input pair of word (w1, w2), where w1 and w2 are words of the pair.
2) Apply each of the above stated techniques to find the score.

D) Score Processing Module:-

As mentioned earlier, that few of the techniques result to similarity while some determine the distance. Dice coefficient, Cosine and Jaccard score gives similarity score in the range of 0 to 1 while Levenshtein and Jaro algorithm provides distance. Hence, in this module, each of the score is converted to similarity (instead of distance).

E) Final Score Module:-

Final Score module will show the final score received from score finder and score processing module. It returns a float value with precision of two decimal values. 0 indicates completely different strings whereas 1 indicates identical strings. The values 0 and 1 are inclusive in the result. Higher number indicates more similarity.

IV. EMPIRICAL RESULTS

In our experiment, we had selected 1000 words retrieved from the Dictionary. This we done for comparing two words using different lexical algorithms. These words are arranged in the same order as their associated Dictionary was downloaded from Google. Score of each set of words was calculated using Dice coefficient, Levenshtein distance, Jaccard similarity, Jaro distance and Cosine similarity and Dice coefficient. We choose these algorithms for comparing on 1000 words results.

In this study of result, we present Dice coefficient on bigrams profiles for word similarity. Dice coefficient measures how similar a set and another set are. It can be used to measure how similar two words are in terms of the number of common bigrams (a bigram is a pair of adjacent(next) letters in the word).For example taking the word "fubar", the set of bigrams would be "fu", "ub", "ba", "ar". Likewise, "foobar" would break down into "fo", "oo", "ob", "ba", "ar". After implementation of Dice coefficient of two words, we found that Dice coefficient give more relevant result when we use bigram compared to trigram.

After implementing of all algorithms and for finding results, we divided score in four types namely Exactly Similar Score (1.0), Near to Similar score (0.75), Near to Dissimilar score (0.50) and Exactly Dissimilar(0) which helped us to finding best algorithm as per their pre-defined values. With the help of different score

types, we can easily find results from system generated output and human judge's results. These score types will categorize the each pairs of words according to their scores into different category.

TABLE I: Results of 10 pairs out of 1000 word pairs

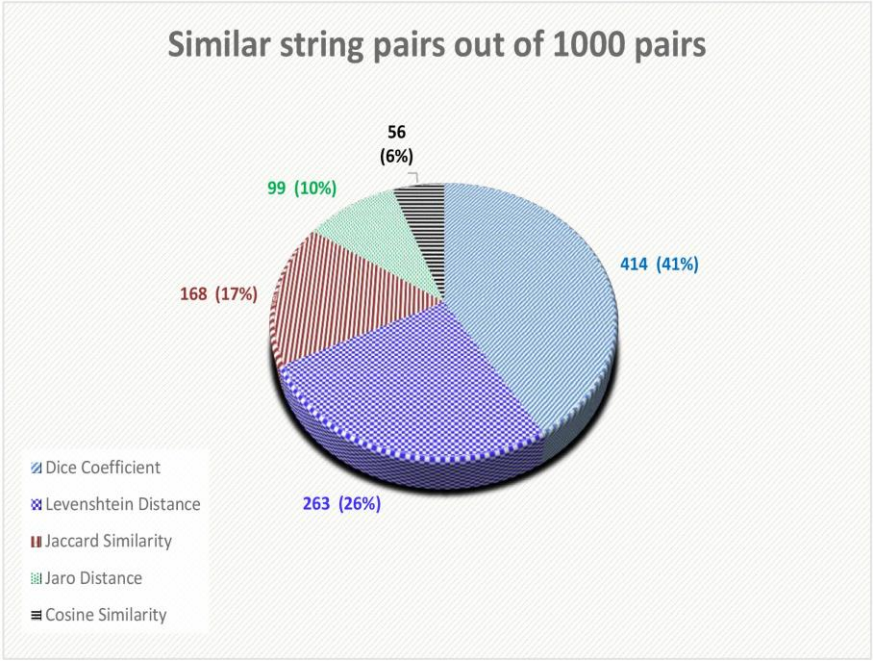
ID	Word pair	Dice Coeff icient	Levenshtein	Jacc ard	Jarro	Cosine	Huma n Score
1	imaging-imaging	1	1	1	1	1	1
2	improbable-imprubebla	0.44	0.7	0.18	0.9	0.92	0.50
3	incarcerate-inceacarenta	0.51	0.50	0.15	0.81	0.94	0.50
4	incandescent-incendasant	0.67	0.67	0.21	0.88	0.93	0.75
5	invokes-invoces	0.67	0.86	0.4	0.9	0.86	0.50
6	Understanding-undarstanding	0.73	0.85	0.44	0.84	0.91	0.75
7	provide-provides	0.77	0.75	0.36	0.87	0.88	0.75
8	illustrator-back	0	0	0	0	0	0
9	antagonise-antegonise	0.78	0.9	0.54	0.93	0.93	0.75
10	bang-beng	0.33	0.75	0.14	0.83	0.75	0.75

It was found that Dice coefficient algorithms gives relevant result of 414 pairs out of 1000 pairs of words when compared with human judges results.

TABLE II: Categories wise algorithms score results.

Name	Dice Coefficient	Levenshtein Distance	Jaccard Similarity	Jaro Distance	Cosine Similarity
Exactly Similar	43	0	0	10	12
Near to Similar	346	248	99	85	43
Near to Dissimilar	18	15	56	4	1
Exactly Dissimilar	7	0	13	0	0

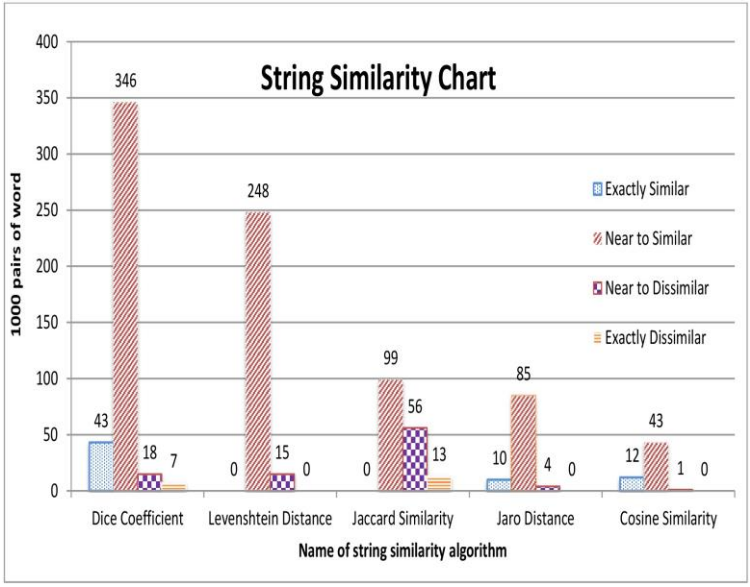
After comparing the results, we found that Dice coefficient gives relevant result from those words which have some same characters, but it fails when user gives those words which have same meaning but different spelling. However, semantic similarity is not within the scope of this paper.



Graph 1. Comparisons of results

Graph 1 and Graph 2 shows the comparison of these algorithms with respect to similar results found from human judges. In graphs, the result is fetched from database where count of each row describes the best algorithm's name against pairs of word. As per these results, we can get accuracy for each algorithm in percentage of 1000 pairs of word by applying below formula:

Result(%)=(Correct_pairs_of_words X 100)/ Total_Pairs



Graph 2. Categorywise comparisons of results

It was also found that such metric can be useful when words are not nouns, and specially proper nouns. This is so because, many a times names are spelled in different way, though they sound same. For example, names Michael and Mikel or Preeti and Priti have different spellings yet sound same. Hence, it is found that those applications that require string similarity should use POS tagging for filtering the words based on their POS tags.

V. CONCLUSION

It was found that similarity scores calculated based on Dice coefficient is more relevant than other algorithms used in comparison. To measure the similarity, mere lexical score is not enough. The semantic features of both the words should also be matched when the word are under comparison. Also, POS tagging is

advisable as such comparisons are applicable and acceptable for verbs, adjectives than proper nouns.

REFERENCE

[1] Yogish, Deepa, T. N. Manjunath, and Ravindra S. Hegadi, "A Survey of Intelligent Question Answering System Using NLP and Information Retrieval Techniques."

[2] Hermjakob, U.; Hovy, E. H.; and Lin, C. 2000. "Knowledge-Based Question Answering." InProceedings of theSixth World Multiconference on Systems, Cybernetics, and Informatics (SCI-2002).Winter Garden, FL: International Institute of Informatics and Systemics.

[3] Feng, D., Shaw, E., Kim, J., & Hovy, E. (2006, January). "An intelligent discussion-bot for answering student queries in threaded discussions." In Proceedings of the 11th international conference on Intelligent user interfaces (pp.171-177).IEEE.

[4] Andrenucci, A., & Sneiders, E. (2005, July). "Automated question answering: Review of the main approaches". In Information Technology and Applications, 2005. ICITA 2005. Third International Conference on (pp.514-519). IEEE.

[5] Hovy, Eduard H., Ulf Hermjakob, and Chin-Yew Lin. "The Use of External Knowledge of Factoid QA."TREC. 2001

[6] Brill, Eric, Susan Dumais, and Michele Banko. "An analysis of the AskMSR question-answering system." Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002.

[7] Radev, Dragomir R., et al. "Mining the web for answers to natural language questions." Proceedings of the tenth international conference on Information and knowledge management. ACM, 2001.

[8] START Natural Language QA system: http://start.csail.mit.edu/index.php

[9] Wikipedia :https://en.wikipedia.org/wiki/ Question_answering

[10] W. Ahmed and B. P, "Question Analysis for Arabic Question Answering Systems", International Journal on Natural Language Computing, vol. 5, no. 6, pp. 21-30, 2016.

[11] V. Thada, and V. Jaglan, "Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm." International Journal of Innovations in Engineering and Technology 2.4, 2013.

[12] L. Cai, G. Zhou, K. Liu, and J. Zhao. "Learning the Latent Topics for Question Retrieval in Community QA.", IJCNLP, vol. 11, pp. 273-281. 2011.

[13] H. Duan, C. Yunb, C. Y. Lin, and Y. Yong, "Searching Questions by Identifying Question Topic and Question Focus." In ACL, pp. 156-164. 2008.

[14] S. P. Abney, "The English noun phrase in its sentential aspect." PhD diss., Massachusetts Institute of Technology, 1987.

[15] V. Shwartz, O. Levy, I. Dagan, and J. Goldberger, "Learning to Exploit Structured Resources for Lexical Inference." In CoNLL, pp. 175-184. 2015.

[16] W. Song, M. Feng, N. Gu, and L. Wenyin. "Question similarity calculation for FAQ answering." In Semantics, Knowledge and Grid, Third International Conference on, pp. 298-301. IEEE, 2007

[17] Stop words list: http://xpo6.com/list-of- englishstopwords/

[18] Part of Speech Information: https://en.wikipedia.org/wiki/Partofspeech_tagging/

[19] Stop words Information: https://en.wikipedia.org/wiki/Stop_words

[20] NLP Information: https://en.wikipedia.org/wiki/ Natural_language_processing/

[21] QA System Information: https://en.wikipedia.org/wiki/Question_answering/

[22] G.Bathla, R.Jindal "Similarity Measures of Research Papers and Patents using Adaptive and Parameter Free Threshold," In International Journal of Computer Applications (0975 - 8887), vol.33- No.5,November 2011.

[23] Maheshkumar B.Landge, Ramesh R.Naik, C. Namrata Mahender,"Measuring Author Impression Using Cosine Similarity Algorithm," In IOSR-JCE, e-ISSN: 2278-0661,p- ISSN: 2278-8727, PP 24-28.

[24] Yufei Sun , Liangli Ma, Shuang Wang, "A Comparative Evaluation of String Similarity Metrics for Ontology Alignment", In Journal of Information and Computational Science 12:3 (2015) 957-964, February 10, 2015.

[25] Maria del Pilar Angeles, Adrian Espino-Gamez, "Comparison of methods Hamming Distance, Jaro, and Monge-Elkan" In The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2015.

[26] Vikas Thada, Dr Vivek Jaglan, "Comparison of Jaccard, Dice, Cosine Similarity Coefficient To Find Best Fitness Value for Web Retrieved Documents Using Genetic Algorithm", In the International Journal of Innovations in Engineering and Technology, SSN: 2319-1058, Vol. 2, 4 August 2013.

[27] https://en.wikipedia.org/wiki/Levenshtein_distance

[28] https://en.wikipedia.org/wiki/Jaro-Winkler_distance

[29] https://en.wikipedia.org/wiki/Cosine_similarity

[30] Hakan Sundblad, "Question Classification in Question Answering Systems", In Diss. Institutionen for datavetenskap, 2007.

[31] Poonam Gupta, Vishal Gupta,"A Survey of Text Question Answering Techniques," International Journal of Computer Applications (0975 - 8887),Volume 53- No.4, September, 2012.

AUTHORS' PROFILE

Mr. Sagar J. Gandhi

He is a student of MCA programme at Shrimad Rajchandra Institute of Management and Computer Applications of UTU, Bardoli.

Mr. Mihirraj M. Thakor

He is a student of MCA programme at Shrimad Rajchandra Institute of Management and Computer Applications of UTU, Bardoli.

Dr. Jikitsha Sheth

She is working as an Assistant Professor at Shrimad Rajchandra Inst. of Management & Comp. Appl., UTU, and Bardoli, India. She is a Ph.D. and has more than 12 years of experience in academics and research in the domain of Natural Language Processing. Her area of interest includes NLP, Machine Learning, Data Science and Analytics. She has under her guidance 2 Ph.D. Candidates. She has organized and attended many workshops and training programmes and has 15 plus research papers published to her credit.

Hariom I. Pandit

He is a student of MCA programme at Shrimad Rajchandra Institute of Management and Computer Applications of UTU, Bardoli.

Hemin S. Patel

He is a student of MCA programme at Shrimad Rajchandra Institute of Management and Computer Applications of UTU, Bardoli.