International Journal of Emerging Science and Engineering (IJESE)
ISSN: 2319–6378, Volume-1 Issue-7, May 2013

# Partially Improved Subsequence Discovery Algorithm for Sequence Matching

**A. D. Pathak, S. J. Karale**

***Abstract— This article describes an Improved technique for the sub sequence discovery algorithm used for natural language processing in question answering system for matching user text input in natural language processing against an existing knowledge base, consisting of semantically described words or phrases. Most common methods & techniques of natural language processing are overviewed and their main problems are outlined. A sequence matching with subsequence analysis algorithm is analyzed and improvements are done which deals with the problems of exact matching,change in custom spelling errors as well as the improvement in the performance metric of the similarity matching.Popular approaches that solve this problem include stemming, lemmatization and various distance functions,sequence matching techniques are analysed to get the better possible technique for solving the problems with higher accuracy. Then the major components of the similarity measure are defined and the computation of concurrence and dispersion measure is presented. Results of the algorithms performance on a test set are then analysed.***

***Index Terms—About four key words or phrases in alphabetical order, separated by commas.***

## I. INTRODUCTION

In recent years Information retrieval becomes most essential task of retrieving the data.ie extracting the data from existing knowledge base. In natural language processing information can be stored in any form and in any language format, the user and researchers are always in hunt of searching and extracting the data or information, which can be used as a resource for enhancing and predicting the future work. For such task researches and users can use Question-answering systems. In Question-Answering system the information to be extracted is provide in the form of query & is searched against the existing knowledge base. For extracting the related knowledge information search algorithms are used in such systems. These algorithms employ different techniques and methodologies to match the users input query against the knowledge base. The techniques may vary according to the applications and the nature of task. The query can be a set finite or infinite collection word or text. The term to searched in the form of query can be in various morphological variation. Popular approaches that are used and are most successful are stemming, lemmatization and various distance functions. In this article we have proposed some improvements in the existing sub sequence discovery algorithm suggested by marko Freme and Milan Ojstersk[1].

**Manuscript received May, 2013.**

**Abhishek.D.Pathak**, Computer science & engineering, Nagpur/YCCE/ MGI,Nagpur,India.

**S.J.Karale**, Asst professor, Computer Technology, Nagpur/YCCE/ MGI,Nagpur,India.

In first phase we have analyzed the popular approaches used in natural language processing for the similarity matching, then the problems are outlined, then improvements are done for overcoming those problems and lastly its performance is analyzed.

## II. SUB SEQUENCE DISCOVERY

As suggested by Marko Freme & Milan Ojstersk[1], Sub sequence discovery algorithm is used to find the text matching from the knowledge base based on similarity matching. This algorithm does not require set of rules for preprocessing of words. This algorithm uses sub sequence from the word or phrase from the query to find out the most similar matching from the knowledge base.

## III. STEMMING

Stemming is a preprocessing step in information retrieval system before indexing and searching. It basically converts morphed words into its root word i.e. stemming gets converted into its root word stem. For reducing the word form from its morphed form it uses the set of rules without considering parts of speech tagging and context of word. The queries fired are segmented and each segment of word is then stemmed and used for searching the document. Helpful Hints

## IV. LEMMATIZATION

In heavily inflected languages the use of lemmatization is preferred. It offers a fast and accurate way of matching user input to morphed instances of a headword but requires exact dictionaries, which have to be build by language experts. A major problem in the process of lemmatization is disambiguation, which occurs when a word or phrase can be transformed into two or headwords. It is most widely being solved with the usage of tree taggers which require large training corpuses and use probability to determine the most suitable headword, which we call a lemma. Building such large collections is very time consuming and requires the aid of language experts. A very large portion of misses in Lemmatization, when being use on heavily inflected languages, is produced from unknown words, such as names, surnames and geographical locations. Those are mostly excluded from dictionaries and tagged corpuses, which makes them nigh on impossible to convert to a lemma. Input error (misspelling) intolerance during lemmatization or tree tagging is in most cases also unaddressed.

Lemmatizing deals with the complex process of first understanding the context, then determining the POS of a word in a sentence and then finally finding the ‘lemma’. In fact an algorithm that converts a word to its linguistically correct root is called a lemmatizer.

A lemma in morphology is the canonical form of a lexeme. Lexeme, in this context, refers to the set of all the forms that have the same meaning, and lemma refers to the particular form that is chosen by convention to represent the lexeme. In computational linguistics, a stem is the part of the word that never changes even when morphologically inflected, whilst a lemma is the base form of the verb. Lemmatizers are difficult to implement because they are related to the semantics and the POS of a sentence. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the lemma [1].

## V. ANALYSIS OF SUBSEQUENCE DISCOVERY ALGORITHM

For analyzing the performance of this algorithm we have implemented this algorithm and checked its performance on the data set of English words. As per given by Marko Freme & Milan Ojstersk[1] the average similarity matching for this algorithm is 86.4 % and since this algorithm is error tolerable and does not require additional rules it is better for the question answering system for semantic analysis. But they have also suggested some enhancements in the working of this algorithm. Such as if lemma is missed then the degree of similarity of this algorithm degrades , if there is change in order of spelling then also its similarity matching degrades as well as the algorithm is not implemented on phrases.

## VI. AIM

As mentioned above the limitations of the sub sequence discovery algorithm we plan to improve the similarity matching by improving and removing the limitations of the sub sequence discovery algorithm. Our approach basically concentrates on the implementation of the algorithm on phrases and change in the order of words of characters

## VII. OUR APPROACH

In this approach we have first done same thing as it is already done in sub sequence discovery algorithm. Firstly we have also used the Longest common subsequence i.e. LCS algorithm for finding out the sub sequences, while finding the sequence we also checked the sequence of order of characters. after getting the sequences of proper order, we then applied the threshold value to filtered out the only those sequences those satisfies the threshold criteria. so at last we get only that output which is most relevant with query , with this also have maintained sequence order in words as well as phrases.

1) As the sequence value < or equal to the threshold value then we remove that sequence form the candidate list and go for the next sequence.
2) As the sequence value becomes equal to the threshold value then that sequence is extracted for the similarity matching.

## VIII. EXPERIMENTAL RESULTS

We made the collection of more than three thousand words and five hundred phrases of English language of average length of 20 letters in word set and average of three words in each phrase i.e. at least of 25 characters and it is then used as the source file as a base for finding out the text to be searched.

In our Analysis we have also done the analysis on spelling errors, In case of spelling errors the algorithm generates the correct prime form from the data set.

We have done the analysis of the algorithm on two criteria

- Occurrence
- Order

We have listed out some example for query to get the analysis of algorithm

| Sr no | Query | Similarity | Avg % |
|---|---|---|---|
| 1 | Abomi | 3/3 | 100% |
| 2 | Evaluation of | 2/2 | 100% |

As mentioned in the above table we have taken two samples one for word and the other for the phrase and in both the example the lemmas are missed but their similarity matching is up to the 100 % as well as the ordering measuring is also 100%. As in the question answering system we fire the query. We have done the same type of analysis atleast on more than 50 examples and we found that in some places where n some words if the part of lemma in the query is present then our similarity matching varies between 50% to 75%, but their also our ordering measure is of 100%. So on the basis of those 50 examples our degree of similarity matching goes up to 93.09 % and in phrases we have done the analysis on the varying factor of threshold as mentioned below.

| Threshold (%) | Concurrence % | Ordering measure |
|---|---|---|
| 100 | 82.30769231 | 100 |
| 83 | 68.89416677 | 100 |
| 70 | 60.65833333 | 100 |
| 50 | 37.23333333 | 100 |
| 28 | 10.95833333 | 100 |
| 21 | 10.08333333 | 100 |

We have implemented all the three algorithms for matching user text input with datasets ,sub sequence discovery algorithm is not implemented on data set of phrase, the aim of our project is to find out the better approach for sequence matching technique .

We have tested the algorithm on word set of 23000 and on 512 phrases so we found that in case of improved SSD algorithm our performance metric gets increased by 7.35 %. Both the algorithms i.e. Sub sequence discovery algorithm & Improved sub sequence discovery algorithm are error tolerable & does not require the support additional rules and dictionary, as well as the algorithm provides the support for flexible sequence order.

## IX. CONCLUSION

In this work we present a set of algorithms that aim to integrate information derived from different knowledge sources in order to enhance the results obtained by Question Answering system. The experiments are promising, showing that the Improved Sub sequence discovery algorithm can exploit the increasing amount of collectively authored, highly heterogeneous, online semantic data, in order to obtain more accurate answers to questions, with respect to a scenario.

As per shown in table we have shown the results of 10 data sets containing of words & phrases. So for Improved sub sequence discovery algorithm the average similarity is increased up to 93.09% & ordering measure is achieved up to 100%, which is better as compared to the previous sub sequence discovery algorithm.

The enhancements suggested by Marko Freme & Milan Ojsteršeko are covered in the improved algorithm. We have tested our algorithm with custom spelling errors. We have also tested our algorithm by changing the order of phrases & it performed well we have also tested our algorithm with various length & order. we have also tested our algorithm on different parameter.

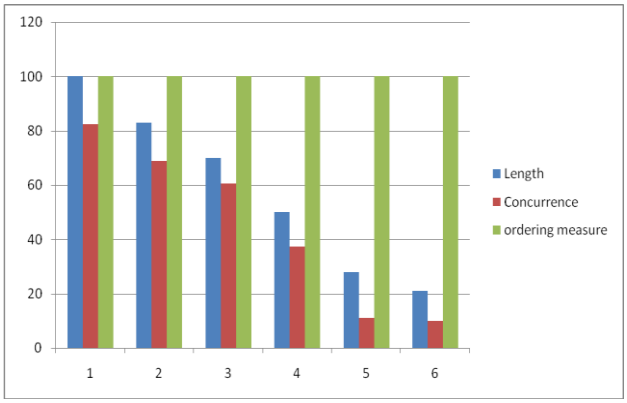


**Fig 1 performance of improved algorithm**

As per the analysis of all the three algorithm i.e. Sub sequence discovery ,Improved sub sequence discovery & stemmer algorithm we found that our Improved algorithm is performing better which is better for the semantic web & search engines.

In the Improved version if the similarity matching is greater than the threshold value then it provides synsets from Word set and the user should select one sense of the synsets offered.

## X. FUTURE SCOPES

The success of a query evaluation depends on a good mapping between the names of relations used in the user's query and names of relations used in the knowledge base. The success of a query evaluation depends on a good mapping between the names of relations used in the user's query and names of relations used in the knowledge base. we propose building a special case of suffix trees best suited for subsequence discovery. Such trees would reduce the time complexity of single sequence comparison against a sequence collection and would allow the development of special algorithm designed to find the most similar match. In course of this work we also plan to evaluate most commonly found subsequences and equip them with statistics and semantics. We plan to extract phrases out of openly available thesauruses such as EuroVoc[6] and add them to our test set. We also plan to insert custom spelling errors and change the word order in phrases to make the test set more representative. With the help of such a test set we plan to improve our algorithms effectiveness by trying out different algorithm parameter values. We also want to test different ways of determining the best sequence match. In our work we simply chose the sequence with the highest similarity measure, but we think that other factors can have a large effect on the most relevant sequence found[1].The algorithm that we have enhanced is better for the question Answering system. Finally, as future work we will explore that how automatically this improved algorithm will generate accurate answers according to the users need. We have also decide to implement this algorithm for Question Answering system.

## REFERENCES

1. Marko Ferme, Milan Ojsteršeko "Sequence matching with subsequence analysis", ISBN: 978-960-474-250-9. Advances in Communications, Computers, Systems, Circuits and Devices.
2. INES ČEH, MILAN OJSTERŠEK "Developing a Question Answering System for the Slovene Language", WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 9, Volume 6, September 2009,ISSN: 1790-0832.
3. Deepa gupta, Rahul kumar yadav, Nidhi sajan, " Improving Unsupervised Stemming by using Partial Lemmatization Coupled with Data-based Heuristics for Hindi" International Journal of Computer Applications (0975 – 8887) Volume 38– No.8, January 2012 .
4. Maria Vargas-Vera, Enrico Motta and John Domingue "AQUA: An Ontology-Driven Question Answering System", AAAI Technical Report SS-03-07.
5. Information Retrieval: Data Structures & Algorithms, edited by William B. Frakes and Ricardo Baeza-Yates.
6. M. Popovic, P. Willett, "The effectiveness of stemming for natural language access to Slovene textual data", Journal of the American Society for Information Science, 43(5), 384–390, 1992.
7. Anjali Ganesh Jivani," A Comparative Study of Stemming Algorithms" Int. J. Comp. Tech. Appl., Vol 2 (6), 1930-1938.
8. " A survey sequence matching & alignment algorithm" by By Jennifer Johnstone.
9. "A Guided Tour to Approximate String Matching by GONZALO NAVARRO, ACM Computing Surveys, Vol. 33, No. 1, March 2001, pp.31–88.
10. [A Fast Generic Sequence Matching Algorithm, David R. Musser Gor V. Nishanov Computer Science Department Rensselaer Polytechnic Institute, Troy, NY 12180 fmusser,gorikg@cs.rpi.edu February 2, 2001.
11. Borut Gorenjak, Marko Ferme, Milan Ojsteršek, "A Question Answering System on Domain Specific Knowledge with Semantic Web Support" INTERNATIONAL JOURNAL OF COMPUTERS Issue 2, Volume 5, 2011.
12. Yajing Zhao,Jing Dong ,senior Member ,IEEE ,and Tu Peng "Ontology classification for Semantic-Web based software Engineering" IEEE Transactions on services computing ,vol 2 no 4 October-December 2009.
13. "Text searching algorithm,volume-1,forward string matching" Borivoj Melichar ,Jan houlab, Tomas Polchar,November 2005.
14. "Udi Manber, Sun Wu. "Fast text searching with errors." Technical Report TR-91-11. Department of Computer Science, University of Arizona, Tucson, June 1991.
15. Udi Manber, Sun Wu. "Fast text search allowing errors." Communications of the ACM, 35(10): pp. 83–91, October 1992, doi:10.1145/135239.135244.
16. A comparison of four pair-wise sequence alignment methods" Nadia Essoussi1 and Sondes Fayech1, published online December 28, 2007.
17. "Solving Sequence Alignment Problem Using Pipeline Approach" by Pankaj Agarwal1 and S. A. M. Rizvi2, BVICAM's International Journal of Information Technology. BIJIT – 2009 Vol. 1 No. 2 ISSN 0973 – 5658.
18. "Method of Fuzzy Matching Feature Extraction and Clustering Genome Data"by Nagamma Patil 1+, Durga Toshniwal 1 and Kumkum Garg 2, IPCSIT vol. 30 (2012) © (2012) IACSIT Press, Singapore.
19. S. Pohorec, M. Verlič, M. Zorman, Domain specific information retrieval system, Proceedings of the 13th WSEAS international conference on computers (part of the 13th WSEAS CSCC multiconference), July 2009, pp. 502-508.