International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 26 Issue 5 – AUGUST 2019.

Sentence Similarity Based Documents Retrieval Using Longest Common Sub-String

Khin Aye Mar[#1]
[#]*Lecturer, Faculty of Computing, University of Computer Studies (Mandalay) Mandalay, Myanmar*

Abstract—Sentence similarity plays an important role in many text-related research and applications such as information retrieval, information recommendation, natural language processing, machine translation and translation memory, and etc. Calculating similarity between sentences is the basis of measuring the similarity between texts which is the key of document classification and clustering. Longest common sub-string (LCS) repeatedly finds and removes the longest common sub-string in the two strings compared, up to a minimum length. Sentence similarity means the similarities between words in different sentences have great influence on the similarity between two sentences. Sentence similarity partially depends on the word similarity. This system will display the similar text of field, areas and other facts in document retrieval.

***Index Terms*— Longest common substring, similarity measure, Tokenization, Stop words**

I. . INTRODUCTION

Words and their orders in the sentences are two important factors to calculate sentence similarity. Sentences connect words and documents in the meaning space of natural language. Similarity analysis between long texts has been widely used in documents retrieval. It considers mainly on the statistical information of keywords in long texts. Sentence similarity plays an important role in text related research and applications. It is closely related to word similarity and document similarity. The LCS can be found by dynamic programming formulation. One can easily show. With a score of 1 for each match and a zero for each mismatch or space, the matched characters in an alignment of maximum value for a LCS [1, 4].

String comparison is a central operation in various environments: a spelling error correction program tries to find the dictionary entry which resembles most a given word, in molecular biology. An obvious measure for the closeness of two strings is to find the maximum number of identical symbols in them (preserving the symbol order) [1, 4]. This, by definition, is the longest common subsequence of the strings. Formally, we are comparing two input strings, X[l..m] and Y[l..n]. A common subsequence (cs) of X[l..m] and Y[l .A] - designated cs(X,Y) - is a subsequence which occurs in both strings. The longest common subsequence (ICS) of strings X and Y, lcs(X, Y), is a common subsequence of maximal length.

The length of lcs(X, Y) is denoted by r(X, Y), or when the input strings are known, by r. The similarity between sentences has great influence on the similarity between documents [2]. Similarity between sentences become increasing important in a variety of applications of natural language process such as text summarization, example-based machine translation, automatic question- answering, information extraction and text clustering.

This paper is organized as follows: section 2 is described Similarity Measure, that it includes algorithm and Tokenization process. Section 3 describes Sentence Similarity Measures and their properties. System Design and Architecture is presented in section 4. Section 5 describes Implementation Results with the figures. Finally, we conclude the paper in section 5.

II. SIMILARITY MEASURES

It has potential applications in many areas such as Pattern Recognition, Data Compression, Word Processing and Genetics. It can be seen as a measure of closeness of two strings as it consists in finding maximum number of identical elements of two strings when preserving the order of element matters. String similarity measures are divided in the literature into three categories: character-based, token-based, and hybrid. In the case of the first two, the similarity is calculated on character and token level respectively. In the case of the hybrid measures, the similarity is first calculated on the character level, and then the obtained scores are used by a token-based metric.

A. Algorithm for Maximal Consecutive LCS

While in classical LCS, the common subsequence needs not be consecutive; in database schema matching, consecutive common subsequence is important for a high degree of matching. It can use maximal consecutive longest common subsequence starting at character 1, MCLCS1 (Algorithm 1) and maximal consecutive longest common subsequence starting at any character n, MCLCSn (Algorithm 2). It present an algorithm that takes two strings as input and returns the shorter string or maximal consecutive portions of the shorter string that consecutively match with the longer string, where matching must be from first character (character 1) for both strings [3, 5].

MCLCS1 (Maximal Consecutive LCS starting at character 1)
input : ri , sj /*ri and sj are two input strings where |ri |= τ, |sj |= η and τ ≤ η */
output: ri /*ri is the Maximal Consecutive LCS starting at character 1 */
τ ←|ri |, η ←|sj |
while |ri |≥ 0 do

```
    If ri ∩ sj then /∗ i.e., ri ⊂ sj = ri∗/
      return ri
    else   ri ←ri \cτ /* i.e., remove the right most character
           from ri */
    end
end
```

B. Tokenization

A text needs to be segmented into words and sentences. This process is called tokenization. Tokenization divides the character sequence into sentences and the sentences into tokens. Not only words are considered as tokens, but also numbers, punctuation marks, parentheses and quotation marks. Each sentence is partitioned into a list of words and removes the stop words. Stop words are frequently occurring, insignificant words that appear in a database record, article or propositions, etc [7].

III. Sentence Similarity Measures

There is a relatively large number of word-to-word similarity metrics in the literature, ranging from distance-oriented measures computed on semantic networks or knowledge-based (dictionary/thesaurus-based) measures, to metrics based on models of information theory (or corpus-based measures) learned from large text collections. If words in two sentences are similar, the two sentences are more possibly similar. If sentences in two documents are similar, the two documents are more possibly similar. The sentence similarity is partially based on the word similarity, and the relations between words should be also considered.

This section presents five measures of statistical similarity between sentences. Sentence similarity based on word set, Sentence similarity based on vector, Sentence similarity based on edit distance, Sentence similarity based on word order and Sentence similarity based on word distance. This system is used the two of the following

1. Sentence similarity based on word set is calculated with the sets of words in two sentences respectively.
2. Sentence similarity based on word order is calculated with the orders between word pairs in the sentences.

The former four sentence similarity measures are symbolic similarity, while the latter two sentence similarity measures are structural similarity. Symbolic similarity between sentences only considers the spelling of words ignoring the meanings of words [2, 5,7].

C. Common Word Order Similarity between Sentences

If the two texts have some words in common, it can measure how similar the order of the common-words is in the two texts (if these words appear in the same order, or almost the same order, or very different order). This system uses it when this system consider the importance of syntactic information by setting its weight factor, w_f to less than 0.5,

Let consider a pair of sentences, P and R has m and n tokens respectively, that is, P = p1, p2, ... , p_m and R = r1, r2, ... , r_n and $n \geq m$. Otherwise, it switch P and R. This system count the number of pi's (say, δ) for which $p_i = r_j$, for all $p \in P$ and for all $r \in R$. That is, there are δ tokens in P that exactly match with R, where $\delta \leq m$. This system remove all δ tokens from P and put them in X and from R in Y , in the same order as they appear in the sentences. So, X ={x_1, x_2, ... , x_δ } and Y ={ y_1, y_2, ... , y_δ }. We replace X by assigning a unique index number for each token in X starting from 1 to δ, that is, X ={1, 2, ... , δ}. Based on this unique index numbers for each token in X , we also replace Y where X = Y. A measure for measuring the common-word order similarity of two texts as:

$$S_0 = 1 - \frac{|x_1 - y_1| + |x_2 - y_2| + .. |x_\delta - y_\delta|}{|x_1 - x_\delta| + |x_2 - x_{\delta-1}| + .. |x_\delta - x_1|} \quad (1)$$

That is, common-word order similarity is determined by the normalized difference of common-word order (the denominator is the largest possible dissimilarity value, the worse order of pair-wise elements in X) [2, 5, 6].

$$S_0 = \begin{cases} 1 - \dfrac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2 - 1} & \text{if } \delta \text{ is even} \\ 1 - \dfrac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2} & \text{if } \delta \text{ is odd and } \delta > 1 \\ 1 & \text{if } \delta \text{ is odd and } \delta = 1 \end{cases} \quad (2)$$

D.Procedure for LCSS

The Longest Common Subsequence Similarity (LCSS) is based on the Longest Common Subsequence (LCS) character-based algorithm. The initial algorithm is transformed into a token-based one. This way the token-level LCS between two sentences is computed. Given two sentences s1, and s2 the computation of the LCSS follows the steps below: Each token in *X* starting from 1 to *δ*, that is, *X* = {1, 2, . . . , *δ*}.

Given two sentences s1, and s2 the computation of the LCSS follows the steps below:

1. Calculation of the LCS at token level:
   LCS-Token-String(s1, s2) = LCS String
2. Calculation of the LCSS at token level as:
   LCSS-Tokens(s1, s2) = Length-token (LCS String) / max (Length-token(s1),Length-token(s2))
3. The computation is done according to a formula similar to the one in step 2: LCSS-Characters(s1, s2) = Length-characters(LCS-String) / max(Length-characters(s1), Length-characters(s2))

LCSS takes values within [0, 1]. 0 indicates that the sentences are completely different and 1 that the sentences are identical.

IV. System Design and Architecture

This system can provide for the new paper publication with similar preceding papers. Editor or user input the title of paper, keywords and abstract section. This system must be tokenized process. Using these token, this system will search similar title in preceding papers database. The outputs are similar title, author name, conference name, and published date with similarity value.

Using Modified LCS, firstly accept input as user desired title in preceding conference papers. This input as sentence, so this system tokenized this sentence and remove the insignificant words using database. This token uses as keyword and the system searches similar word of preceding

title in database [2,5, 6].

E. System flow diagram

Figure1 illustrates the Major process of redundant paper searching system. This system consists of four processes:

Stop word Removal, Token Recognization, Maximal Consecutive LCS and Modify LCS algorithm.

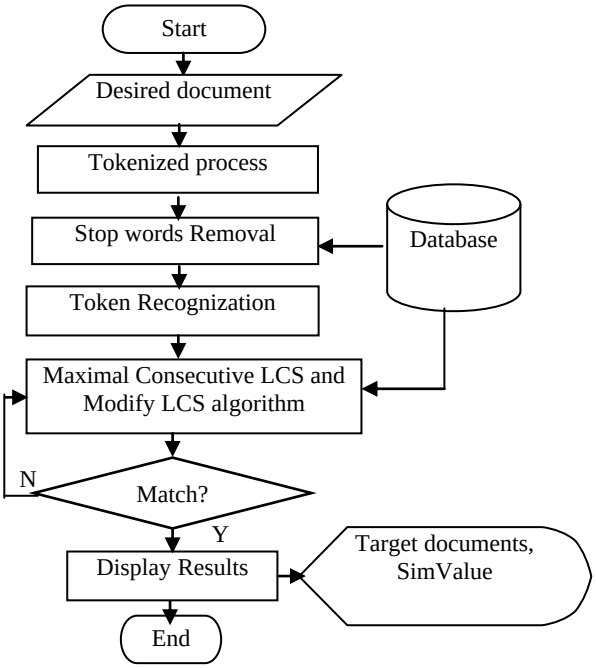

Figure 1 System flow diagram

The process of stop word removal is to remove articles, prepositions, conjunctions, frequent words and insignificant words of the searching titles. Firstly, the input search title was tokenized. And then, this process removed the stop words according to the Stop Word array. The stop word removal process in this system generates the string that has been removed stop words. The output of this process is the input of the tokenized process. In this process, the variable N is the Length of Token array. And the variable K is the Length of the Stop Word array.

In MCLCS algorithm, his process accepts the Search Array as an input. And then, the process put all documents from Database into Title array. This process creates the integer array Title-Count array to store the count of the number of the similar words in the title. This process checks each element in the title array to find the similar words of the input title. Finally, maximum consecutive longest common substring algorithm generates the all similar titles in database with the input title by similar word count. In this process,

- the variable N is the length of the Title array
- the variable K is the length of the Search Arrray
- the variable L is the length of the Title Array
- the variable S is the length of the elements of the Search Array
- the variable T is the length of the elements of the Title array

F. Sample Stop words

TABLE 1. SAMPLE STOP WORDS

Sr. No	Categories	Descriptions
1	a, an, the	articles
2	in, or, on, of, into, by	prepositions
3	and, but, for, between	conjunctions
4	using, implementation, simulation	frequent words
5	system, method, algorithm, approach	Insignificant words

In this system, stop word removal process can take place for accurate result. In stop word table, articles, prepositions, conjunctions, frequent words and insignificant words are included.

V. IMPLEMENTATION RESULTS

Paper searching system is implemented by using web technology and C# programming language. This system can display the similar methods, fields, keywords with similarity value and similar word count. The title with large similarity count is display on top of page with ascending order.

Firstly, system will remove stop words to calculate similarity measure as shown in figure 2.

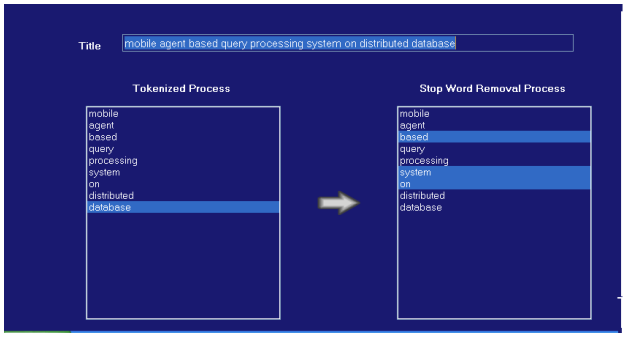

Figure 2 Stop words Removal process

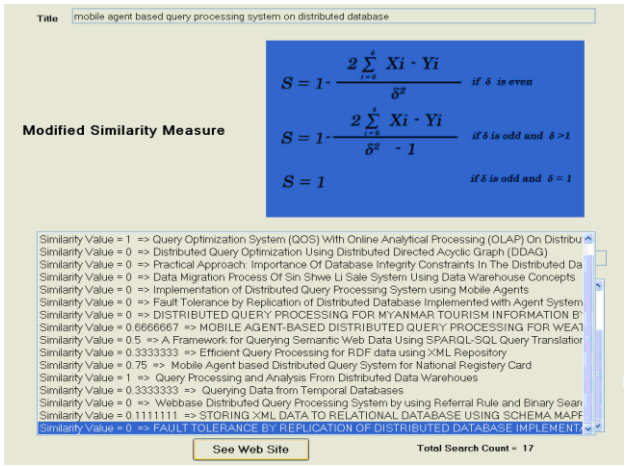

Figure 3 Result using Modified Similarity Measure

This system is processed tokenization and stop words removal using stop words table. These tokens are used as key words in this system. Using keywords in input title, this system is applied maximum character longest common substring algorithm to search similar titles. If the similar tiltes

are found by samilar words, it will displayed paper title with similar count number in figure 4. The keywords of this example are mobile, agent, query, processing, distributed and database.

If user want to know the more similar titles and same order of desired title and search title, user can use word order similarity that is Modify similarity measure for each sentenses. Using three type of formula in modified similarity measure, this system will calculate the similarity value for samilar sentenses. This process is calculate the similarity value in each serched sentense.

This system can search many similar titles with only single words as shown in figure 3. This result can support for redundant title for students and supervisors.

VI. Conclusion

This system can support for paper publication system. This system will use longest common substring to get common texts. These common texts will be methods, algorithms and applications in preceding papers. Administrator for paper publication can use this system to provide redundant paper for publication. This system uses LCS to obtain the similarity of the query and the title of the documents to find more similar documents in the retrieved documents.

Semantic similarity is computed based on the two semantic vectors. An order similarity is calculated using the two order vectors. Finally, the sentence similarity is derived by combining semantic similarity and order similarity[1].

Measures of text similarity have been used for a long time in applications in natural language processing and related areas. This system considers for finding a longest common subsequence LCS for two strings S1 and S2 such that common string is a subsequence of the solution LCS. An effective method to compute the similarity between texts or sentences has many applications in natural language processing and related areas such as information retrieval and text filtering.

Other string matching methods such as Edit distance, n-gram and Dynamic Programming techniques can be used in this application. This system can extend the performance of information retrieval. To try to give a more qualitative view of the results compared to the results with other methods.

References

[1] A. Islam and D. Inkpen. “Semantic text similarity using corpus-based word similarity and string similarity”. ACM Transactions on Knowledge Discovery from Data (TKDD), 2(2):1–25, 2008.
[2] Apostolico, C. Guerra. 1987. The Longest Common Subsequence Problem Revisited, Algorithmica, pp. 315-336.
[3] F. Y. L. Chin, C. K. Poon. 1990. A fast algorithm for computing longest common subsequences of small alphabet size, Journal of Information Processing, v.13 n.4, pp. 463-46.9
[4] G. Navarro. 2001. A Guided Tour to Approximate String Matching, ACM Computing Survey, Vol. 33, No. 1, pp. 31-88.
[5] L. Bergroth, H. Hakonen, T. Raita. 2000. A survey of longest common subsequence algorithms, 7th International Symposium on String Processing Information Retrieval, pp. 39–48.
[6] Mark Gondree and Payman Mohassel. Longest common subsequence as private search. In Proceedings of the 2009 ACM Workshop on Privacy in the electronic society (WPES 2009), 2009.
[7] Pedersen, T. and Patwardhan, S. and Michelizzi, J. “WordNet::Simil relatedness of concepts”, Association for Computational Linguistics, 2004.