IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 2, Ver. IV (Mar – Apr. 2015), PP 32-41
www.iosrjournals.org

An Enhanced Suffix Tree Approach to Measure Semantic Similarity between Multiple Documents

A.Kavitha[1], Dr.N.Rajkumar[2], Dr.S.P.Victor[3]
[1](Research Scholar, Manonmaniam Sundaranor University, Tirunelveli, India,)
[2](Dept. of M.E. S/w Engg., Professor & Head, Sri Ramakrishna Engineering College, India,)
[3](Dept. of MCA, Professor & Head, St. Xavier College, Palayamkottai, Tirunelveli, India,)

Abstract: *Semantic Similarity is a concept whereby the set of documents are measured to find the likeliness of their meaning content. Document Similarity is the process of Computing the Semantic Similarity between Multiple Documents Using Similarity measures. In this paper, the document similarity has been applied to compute the pair wise similarities of documents based on the Suffix Tree Document (STD) model. Documents are pre-processed initially. Data Preprocessing can be done to increase the efficiency of the Similarity values. The pre-processed phrases are inserted in Suffix tree. A Suffix tree is a data structure that presents the suffixes of a given string in a way that allows for a particularly fast implementation of much important string operation. The suffix substrings are selected as the phrases to label the edges of the suffix tree. Internal nodes represents phrases that shared by Multiple Documents. The similarity of two documents can be defined as the more internal nodes shared by the two documents. Suffix tree can be used to solve the exact matching problem in linear time. Document similarity naturally inherits the term tf-idf(Term frequency and inverse Document frequency) weighting scheme in computing the document similarity with phrases. Tf-Idf method has been used to calculate the weight of Internal nodes of the suffix tree, where internal nodes are the nodes that has been shared by multiple documents. Cosine, Dice and Hellinger measures applied to find the pair wise similarity based on the weight of each internal node of the suffix tree.*

Keywords: *Semantic similarity, Similarity measures, Document similarity, Suffix tree and Tf-idf scheme.*

I. Introduction

Semantic similarity is a domain whereas a set of documents within lists are assigned a metric based on the likeness of their meaning content. The document similarity plays a vital role in the field of information retrieval using Clustering technique [11][7]. The main goal of the system is to compute the semantic similarity between multiple documents. The system involves by getting the several documents as input from the user to find the similarity between various documents based on different similarity measure. The document preprocessing denotes the Stop words removal, Case conversion and Special characters removal. The phrases are extracted from the document to construct the suffix tree and labeled to edges of the nodes of the suffix tree [1][10]. A Suffix tree is a data structure that presents the suffixes of a given string in a way that allows for a particularly fast implementation of many important string operations [14][9]. The term frequency Tf-Idf method is used to calculate the weight of internal nodes of the suffix tree, where internal nodes are the nodes that have been shared by multiple documents. Cosine similarity measure, Dice Coefficient and Hellinger measures are used to find the pair wise similarity based on the weight of each internal node of the suffix tree [5][7]. Document similarity is shown as values and the values must be between 0 and 1. The value 1 implies the absolute similarity and 0 implies both the documents are not similar.

1.1 Semantic Similarity

Semantic similarity measures can be classified into pair wise similarity and group wise similarity measures. The Pair wise similarity measures functional similarity between two instances by combining the semantic similarities of the concepts they represent. The group wise semantic similarity measure calculates the similarity directly by not combining the semantic similarities of the concepts they represent.

Semantic similarity is mostly used approach and associated with several applications to determine similarity [15]. The similarity measures are used in conjunction with corpus system to retrieve all kind of information and also it helps to retrieve information in web [3][4][8].

1.2 Data Preprocessing

The data pre-processing in an existing consist of three phase namely, special character removal, stop words removal and case conversion. The data pre-processing helps to minimize the document size and comparison time. In the first phase, list of 32 special characters are removed from all the documents [1]. The few special characters are shown in fig. 1.

!, @, #, $, %, ^, &, *, (,),-,=,+,_,[,], ;,:,|,<,>,?,/,`,~ , , \

Figure 1. Special characters list

The second phase is a removal of stop words and it eliminates over all 256 stop word list from all the input documents. The list of stop words is presented in fig. 2.

a, an, the , is , are , there, who, what, when, how, much, this, that,.. etc.

Figure 2. Stop Words List

The third phase is case conversion, it converts entire document from uppercase to lower case.

Example

The data preprocessing process has been illustrated to the following document as in fig. 3 and fig. 4.

Computer science or Computing science (abbreviated as CS or CompSci) is the scientific and practical approach to computation and its applications. A computer scientist specializes in the theory of computation and the design of computational systems.

Figure 3. Document1

Computer science Computing science abbreviated CS or CompSci scientific practical approach computation applications computer scientist specialize theory computation design computational systems

Figure 4. Preprocessed documents

II. Related Work

Hung Chim and Xiaotie Deng,(2008) proposed a method to compute document similarity. The main objective of their work was to find a phrase-based document similarity to compute the pairwise similarities of documents based on the Suffix Tree Document (STD) model. By mapping each node in the suffix tree of STD model into a unique feature term in the Vector Space Document (VSD) model, the phrase-based document similarity naturally inherits the term tf-idf weighting scheme in computing the document similarity with phrases [1].

Elias Iosif and Alexandros Potamianos presented a Web-based metrics that compute the semantic similarity between words or terms and compared with the state of the fine art. Starting from the fundamental assumption that similarity of context implies that similarity of synonym and relevant Web documents were downloaded via a Web search engine and the contextual information of words of interest can be compared (context-based similarity metrics). In addition, the proposed unsupervised context-based similarity computation algorithms seems to be competitive with the state-of-the-art supervised semantic similarity algorithms based on language-specific knowledge resources [2].

Chen et al. proposed Story Link Detection systems that determines whether two stories are about the same events or links which are usually based on the cosine similarity measure between two stories. This work presents a method for increasing the performance of a link detection system by using a variety of similarity measures and using source-pair specific collective information. The various similarity procedures such as cosine, Hellinger, Tanimoto and clarity, both alone and in combination have been used [5]. Jaz et al presented to methods to learn semantic similarity between documents. One method is based on document similarity and other approach based co-occurrence information [13].

Sheetal A et al. presented a method to compute similarity between words through web documents. Semantic similarity measures play an important role in the extraction of semantic relations. It uses the web based metrics to compute semantic similarity between words or terms and also compares with the state-of-the-art. Similarity measures proposed in this work based on the five different association measures in retrieval of information that is normal matching, Dice, Jaccard, Overlap, and Cosine coefficient. The performance of these methods has been evaluated using Miller and Charle's benchmark dataset [6].

Anna Huang implemented a method to analyze the effectiveness of similarity measures in partitional clustering for text document datasets. This proposed approach utilized the standard K-means algorithm and report the results on several text document datasets and five distance/similarity measures that have been most commonly used in text clustering [7]. Hsun and yau presented the work of cross language retrieval using semantic similarity measures. They applied fuzzy models to represent the document and used similarity approaches to retrieve information [12].

III. Proposed Work

The proposed system includes four major methods to compute an efficient similarity between document work namely Data Preprocessing, Suffix tree, Node Weight calculation and Similarity Measures. The proposed work includes the stop nodes removal that is removal of symbols, Stop words and Case Conversion. Phrases can be extracted from the pre-processed data. Each internal node has at least two children and each edge

is labeled with a nonempty sub-string of a document known as a sentence. Every leaf node in the suffix tree designates a suffix sub-string of a document; each internal node shows a phrase shared by at least two suffix sub-strings. The similarity of two documents is defined as the more internal nodes shared by the two documents, the more related the documents be likely and includes different similarity measures to show the different between the range of the similarity and the flow of proposed the similarity measures includes three different measures such as Hellinger, Jacard and Dice coefficient. The proposed work is shown in the Fig. 5.

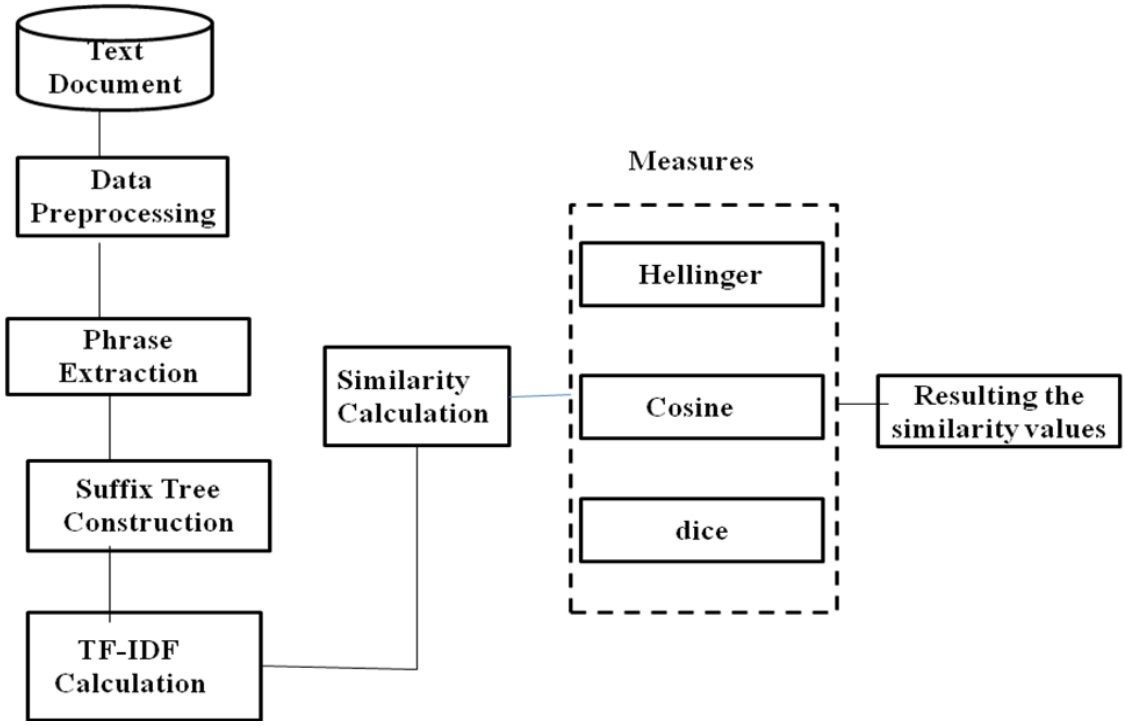


Figure 5. Proposed system Architecture

3.1 Suffix Tree

A tree-like data structure for solving problems contains strings which allow the storage of all sub-strings of a given string in linear space. Each internal node, except root node, contains minimum two children and every edge is labeled with a nonempty sub-string of S. Suffix tree is considered to be one of the well-known full text index data structures. It has been studied for decades and is used in many algorithmic solutions and practical applications. The necessary steps to be followed to construct suffix tree consists of extracting the phrases form the preprocessed document and each edge is labeled with a nonempty sub-string of a document called a phrase. There are three kinds of nodes in the suffix tree: the leaf nodes, root node and internal nodes. Every internal node represents a common phrase shared by at least two suffix sub strings. The similarity of two documents is defined as the more internal nodes shared by the two documents, the more exact documents it should be. The leaf nodes can be called as terminal nodes. Each node in the suffix tree, except terminal nodes and the root node, either an internal node or a leaf node represents a nonempty phrase that appears in at least one document in the data set. The similar phrase may exist in various edges of the suffix tree. The suffix tree of a document set is a compact trie containing all suffix sub-strings of the documents in the data set. During the suffix tree construction, the root node is the initial node and the parent of all other nodes. All other nodes are created and stored in a hierarchical order to follow their LCP nodes, respectively. In our contribution, all the child nodes of the root node are defined as first-level nodes of the suffix tree, the child nodes of the first-level nodes as second-level nodes and so on.

To build a suffix tree, the naive and straightforward method searches each suffix sub-string of the document to all suffix sub-strings which already exist in the tree and finds a position to insert it. The time complexity of building the suffix tree for a document of m words is O (m^2).

Example

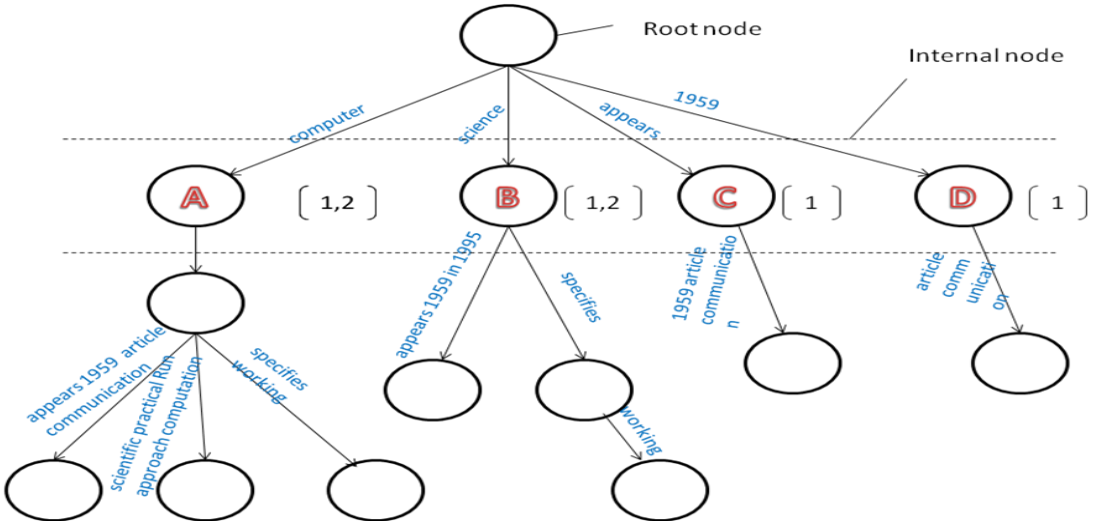
Consider the two Documents:

Document 1

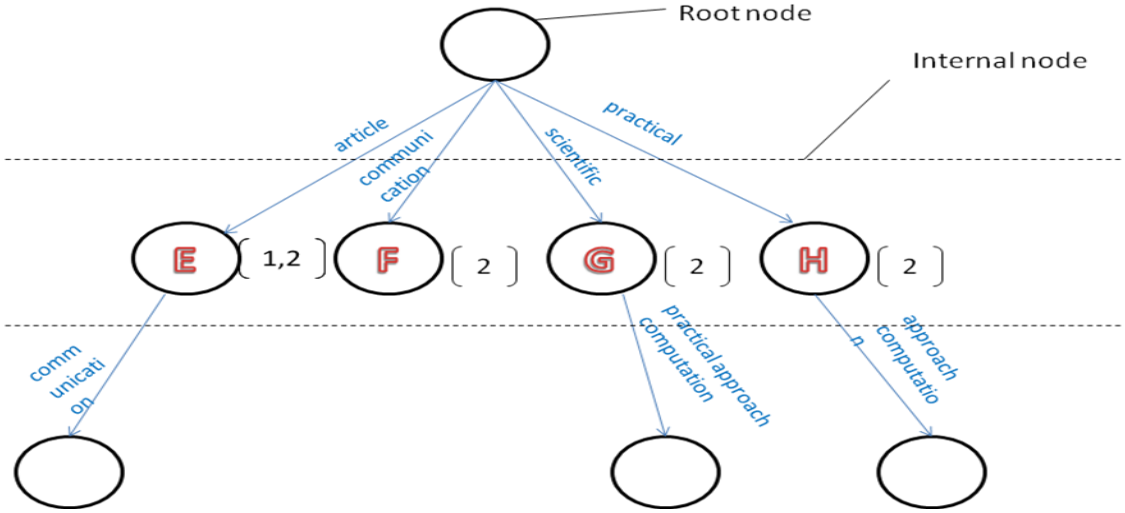
Computer science computing science abbreviated cs or compsci scientific practical approach computation article computer scientist specializes theory computation design computational systems

Document 2

Computer science appears 1959 article communication Human interaction considers challenges making computers computations useful usable universally accessible humans



Cont..



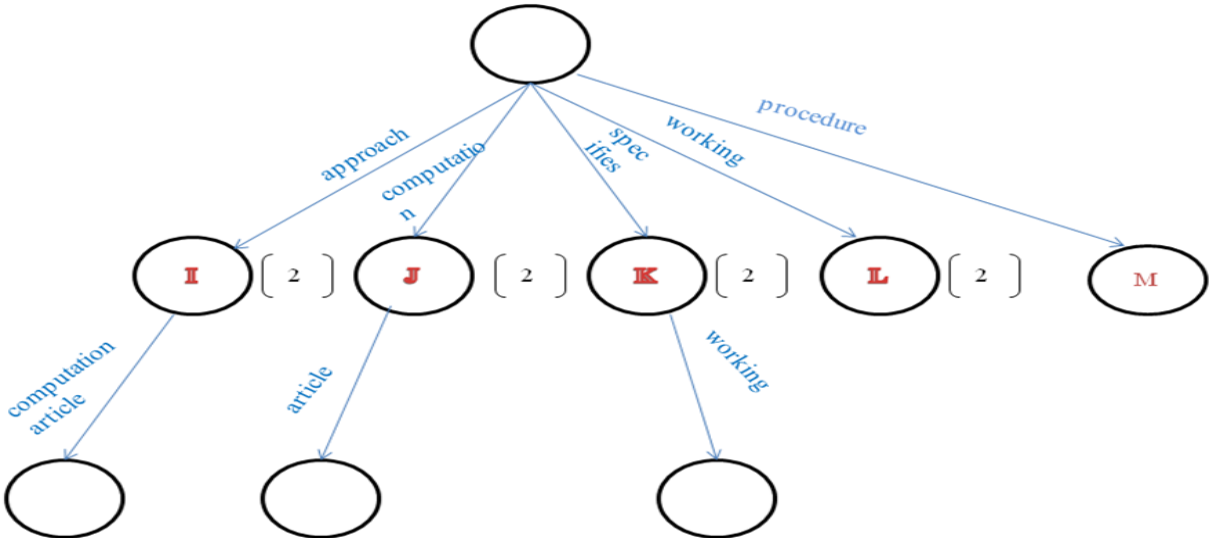


Figure 6. Suffix tree

Nodes shared in the above Suffix tree are A, B and E.

3.2 Weight Calculation

Weight of the node can be calculated using TF-IDF weighting scheme, where tf- refers term frequency and df- refers inverse document frequency , is a numerical statistic which reflects how important a word to a document in a set. It is frequently used as a weighting feature in information retrieval and text mining.

The tf(t,d) represents the number of times that term t occurs in document d.

The inverse document frequency (idf) is a measure of whether the term is common or rare across all documents. The Idf is obtained by dividing the total number of documents by the number of documents containing the term.

The node weights in the documents to be calculated using equation (1).

$$\mathbf{d=\{w(1,d),w(2,d),\ldots\ldots w(m,d)\}} \quad (1)$$

Where w=weight and m=number of terms. The weight of the term can be calculated using equation (2).

$$\mathbf{w(i,d)=(1+\log tf(I,d).\log(1+N/df(i))} \quad (2)$$

Where, tf(i,d),is the frequency of the i^{th} term in the document, and df(i) ,is the number of Documents containing the i^{th} term and N refers number of Documents**.**

Example: Calculating the weight of the internal nodes shared by multiple Documents.

Internal nodes Shared by Multiple Documents in fig. 6 are Node A,B and E.

Calculating the Weights

w(a,1)=w(computer,doc1)=(1+log tf(computer,doc1)).log(1+N/df(computer))

→ tf(computer,doc1) = 1
→ df(computer) = 2
→(1+log 1).log(1+2/2)
→(1+0).log(1+1)
→(1).(0.693)
→ 0.693

W(Computer,doc1)=0.693

w(B,doc1)=w(science,doc1)

→(1+log tf(science,doc1).log(1+N/df(science))
→tf(science,doc1)=1
→df(science)=2
→(1+log 1).log(1+2/2)

→(1+0).log(1+1)
→0.693

W(Science,doc1)=0.693

Similarly, calculate the value of node B and E with respect to Document 1 and Document 2.

Node weight table is constructed from the above calculation as shown in table 1.

Table 1. Node Weight Table

NODE	DOC 1	DOC 2
A	**0.693**	**0.693**
B	**0.693**	**1.173**
E	**0.693**	**0.693**

3.3 Similarity Measures

Similarity Measure is a measure which computes the semantic similarity of the documents using similarity values and the similarity method can represents the similarity between multiple documents. The measure reflects the degree of closeness or likeliness of two documents. All similarity measures should map to the range [-1, 1] or [0, 1] , 0 or -1 minimum similarity and 1 shows maximum similarity. The proposed approach has been applied three different similarity measures: Cosine similarity, Dice Coefficient and Hellinger Measure. There is a large number of similarity measures proposed in the survey, since the finest similarity measure is not exist.

3.3.1 Cosine Similarity

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The resulting similarity ranges from −1 to 1 and 0 usually representing autonomy, and values in between represents intermediary similarity or dissimilarity. In the case of similarity measure, the cosine similarity of two documents may be series as of 0 to 1, because the term frequencies may not be negative.

$$\text{Cosine Similarity} = \frac{dx.dy}{|dx|.|dy|} \rightarrow \frac{\sum_{i}^{m}{}_{=1}\, x_i.y_i}{\sqrt{\sum_{i=1}^{m} x_i^{\,2} y_i^{\,2}}} \quad (3)$$

Where dx and dy are the Documents

dx={x1,x2,x3……xn} and dy={y1,y2,y3…..yn}, x_i and y_i is the weight of corresponding nodes and m and n are the number of internal nodes.

Doc 1 ={A,B,E}
Doc 2 ={A,B,E}
where, x is A, y is B and z is E.

$$\text{Cosine} = \frac{(x1*x2)+(y1*y2)+(z1*z2)}{(x1^2+y1^2+z1^2)^{1/2}\,(x2^2+y2^2+z2^2)^{1/2}}$$

$$= \frac{(0.693*0.693)+(0.693*1.173)+(0.693*0.693)}{((0.6993)^2+(0.693)^2+(0.693)^2)^{1/2}.((0.693)^{2+}(1.173)^2+(0.693)^{2)1/2}}$$

$$= \frac{1.7732}{(1.4406)^{1/2}.(2.257)^{1/2}}$$

$$= \frac{1.7732}{(1.200)(1.502)}$$

$$= 0.98$$

Cosine Similarity for the Document 1 and Document 2 is 0.98.

3.3.2 Dice Coefficient

Dice coefficient determines how similar a set and another set are. It can be applied to measure how similar two Documents are in terms of number of common bi-grams. Dice coefficient is mainly used for comparing the similarity of two Documents and it uses statistic to compute the similarity of two samples.

$$\text{Dice coefficient} = \frac{2\,A.B}{|A|+|B|} = \frac{2(\sum x_i y_i)}{\sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} y_i^2} \quad (4)$$

Where A and B are the Documents

Doc 1 ={A,B,E}
Doc 2 ={A,B,E}

→ $$\frac{2\,((0.693 * 0.693)+(0.693*1.173)+(0.693 * 0.693))}{((0.6993)^2+(0.693)^2+(0.693)^2)+((0.693)^{2+}(1.173)^2+(0.693)^{2)}}$$

→ $$\frac{2(1.7732)}{(1.4406)+(2.257)}$$

→ $$\frac{3.5464}{3.6976}$$

→ 0.956

Dice coefficient for Document 1 and Document 2 is **0.956**

3.3.3 Hellinger Distance

Distance between probability distributions is called as Hellinger distance. The Hellinger distance is closely associated to the total variation distance. For example, both distances define the same topology of the space of probability measures, but it has several technical advantages derived from properties of inner products.

$$\text{Hellinger} = \frac{\sum xi\ yi}{(\sum i=1m\ \ xi2\ +\sum i=1m\ yi2\ )\ -\sum i=1n\ \ (xi-yi\ )} \quad (6)$$

→ $$\frac{((0.693 * 0.693)+(0.693*1.173)+(0.693 * 0.693))}{((0.6993)2+(0.693)2+(0.693)2)+((0.693)2+(1.173)2+(0.693)2)\ -\ ((0.693*0.693)+(0.693*1.173)+(0.693 * 0.693))}$$

→ $$\frac{(1.7732)}{((1.4406)+(2.1334))-1.7732}$$

→ $$\frac{1.7732}{3.574-1.7732}$$

→ 0.984

Hellinger Distance for Document 1 and Document 2 is **0.984.**
The comparison of two documents using Cosine, Dice and Hellinger distance has shown in table 2.

Table 2. Comparison table of two different Similarity measures

Measures	Similarity Values
COSINE	0.99
DICE	0.956

Hellinger	0.984

IV. Performance Evaluation

In order to evaluate the performance of the proposed system, it has been developed using NetBeans IDE version 7.2 for UI and computing the values and Microsoft Access for database. The set of standard data from www.Wikipedia.com source and also some dataset from www.uc.dataset.org has been collected and employed to the evaluation of the system.

This system gives the document similarity values between 0 and 1. Multiple documents that are any number of documents can be compared to get the similarity values using Cosine, Dice and Hellinger measures. The preprocessing method reduces the complexity of the suffix tree and increases the accuracy of the Similarity measures by eliminating irrelevant terms and symbols as node. The String matching and term weight can be easily calculated using Suffix Tree procedure. Fig, 7 describes the size of suffix tree growth linearly to the size of documents. The line shows the number of internal nodes in suffix tree against the number of nodes exist in every document.

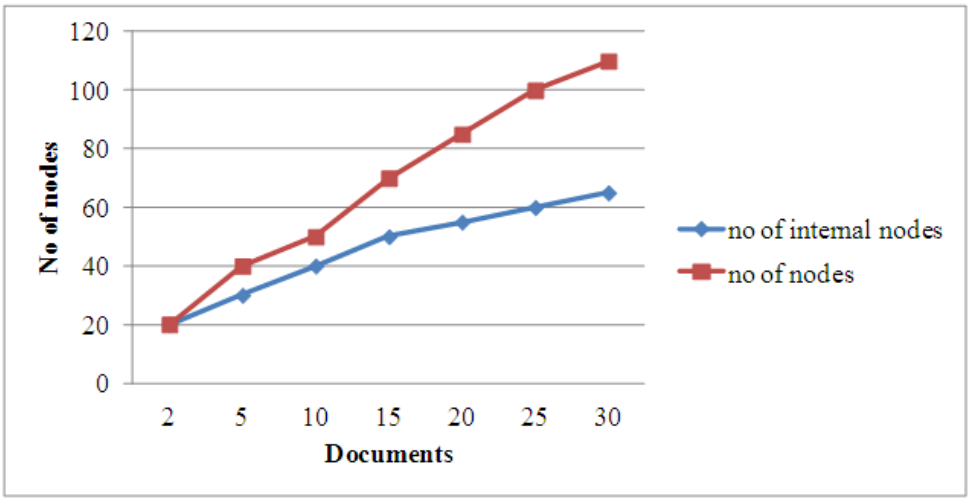


Figure 7. The size of suffix tree scales linearly to the size of document

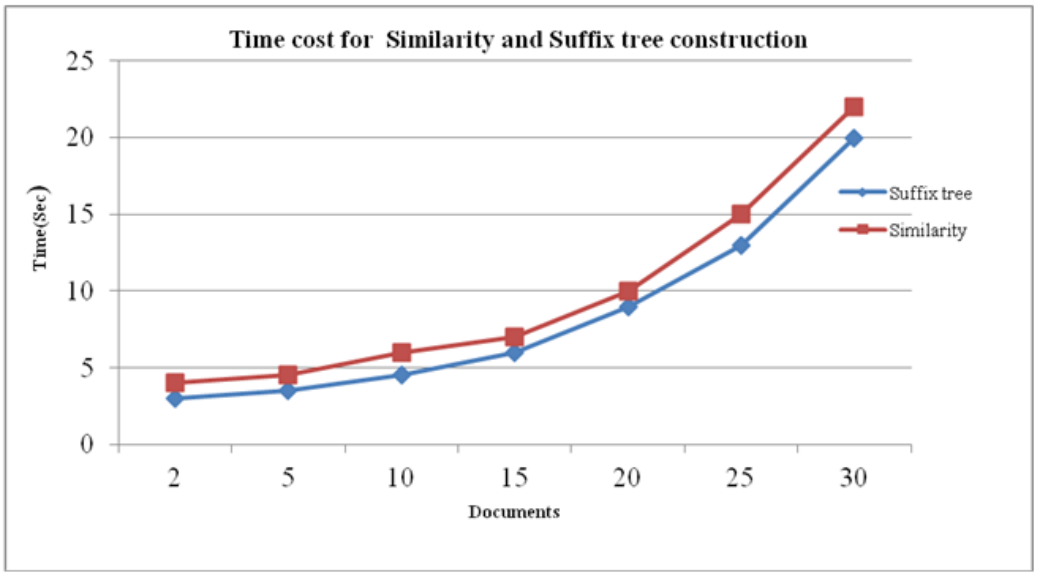


Figure 8. Time cost for Similarity and suffix tree construction

The fig. 8 shows the time required to construct the suffix tree and similarity calculation. The time gradually increases with the number of documents in the system. The comparison of similarity result from the Hellinger, Cosine and Dice is presented in fig. 9.

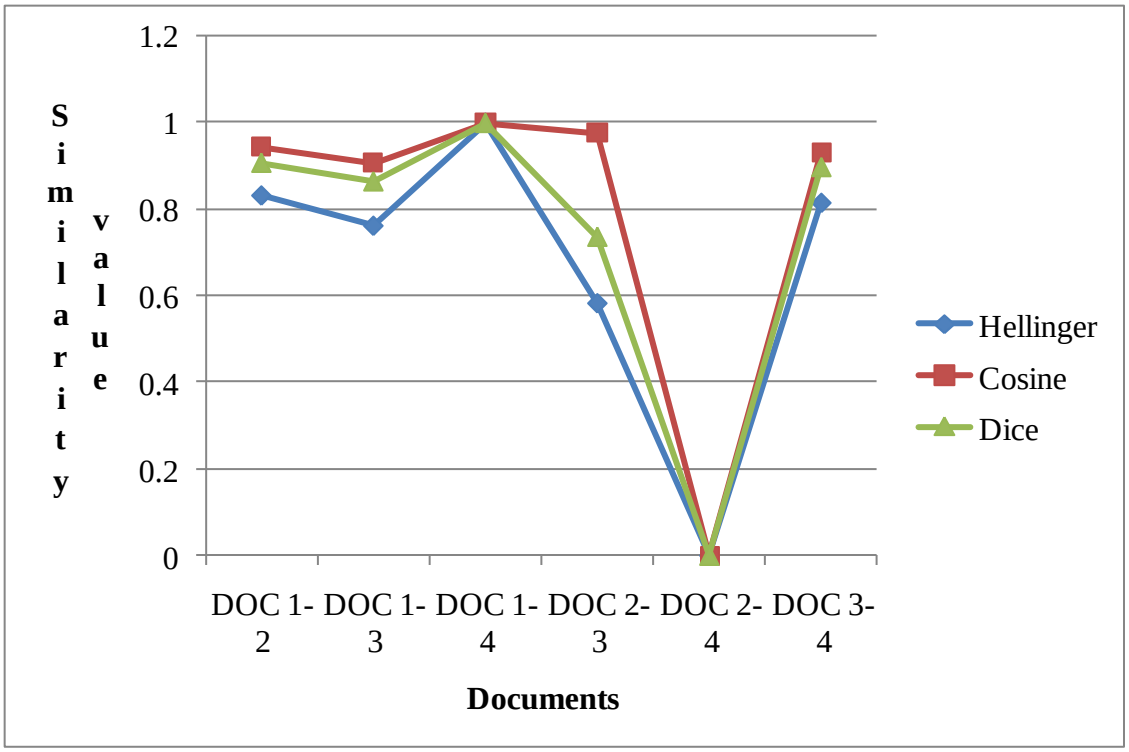


Figure 9. Comparison of different similarity measures

V. Conclusion And Future Work

The paper successfully computes the similarity of multiple documents and gives the similarity in values. The concept of the suffix tree and the new document similarity are quite simple, but the implementation of these approaches is little bit complicated. To improve the performance of the document similarity, we investigated the STD model in both the theoretical data structure analysis and the clustering algorithmic optimization. As a result, the efficiency of the new document similarity approach has been proven in our experiments on large document dataset. The phrases tf-idf weights has been used in computing document similarities and proven to be very effective in documents similarity. Our work has reported a successful approach to extend the usage of tf-idf weighting scheme. The term tf-idf weighting scheme is suitable for evaluating the importance of not only the keywords but also the phrase in document clustering. The replacement of Suffix tree with Enhanced suffix Arrays improves the space efficiency. Enhanced suffix arrays satisfy the algorithm of the suffix tree to overcome the space and time complexities. The future scope of the system will focus on accepting all types of documents to determine the similarity.

References

[1]. Hung Chim and Xiaotie Deng, "Efficient Phrase-Based Document Similarity for Clustering" IEEE Transactions On Knowledge And Data Engineering, Vol. 20, No. 9, pp. 1217-1229, 2008.
[2]. Elias Iosif, Alexandros Potamianos, "Unsupervised Semantic Similarity Computation between Terms Using Web Documents" IEEE Transactions On Knowledge And Data Engineering, Vol. 22, No. 11, pp: 1637-1647, 2010.
[3]. Angelos Hliaoutakis, et al. , "Information Retrieval by Semantic Similarity" , in. International Journal on Semantic Web & Information Systems, Vol.2, No.3, pp.55-73, 2006.
[4]. Giannis Varelas et al., "Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web", Proc. of the 7th ACM International workshop on Web information and Data Management , pp. 10 -16, 2005.
[5]. Francine Chen, Ayman Farahat, Thorsten Brants, "Multiple Similarity Measures and Source-Pair Information in Story Link Detection", Proc. of Human Language Technology Conference, pp. 313-320, Chicago, 2004.
[6]. Sheetal A. Takale, Sushma S. Nandgaonkar , "Measuring Semantic Similarity between Words Using Web Documents", International Journal of Advanced Computer Science and Applications, Vol. 1, No.4, pp.78-85, 2010.
[7]. Anna Huang, "Similarity Measures for Text Document Clustering", Computer Science Research Student Conference, pp.49-56, New Zealand, 2008.
[8]. Danushka Bollegala, Yutaka Matsuo and Mitsuru Ishizuka, "A Web Search Engine-Based Approach to Measure Semantic Similarity between Words", Knowledge And Data Engineering, Vol. 23, No. 7, pp.977-990, 2011.
[9]. D.S. Sven Meyer zu Eissen and M. Potthast, "The Suffix Tree Document Model Revisited," Proc. Fifth Int'l Conf. Knowledge Management (I-Know '05), pp. 596-603, 2005.
[10]. Mohamed Ibrahim Abouelhoda, Stefan Kurtz and Enno Ohlebusch, "Replacing suffix trees with enhanced suffix arrays", Journal of Discrete Algorithms Vol.2, No.1, pp.53–86, 2003.
[11]. Behnam Hajian and Tony White, "Measuring Semantic Similarity using a Multi-Tree Model", Proc. of 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, pp. 7–14, Spain, 2011.

[12]. Hsun-Hui Huang and Yau-Hwang Kuo, "Cross-Lingual Document Representation and Semantic Similarity Measure: A Fuzzy Set and Rough Set Based Approach", IEEE Transactions on fuzzy systems, vol.18, no.6, pp.1098-1111, 2010.
[13]. Jaz Kondola,John Shawe-Taylor,Nello Cristianini, "Learning Semantic Simlarity", Proc. of Neural Information Processing Systems, vol.15, pp.657-664, Canada, 2003.
[14]. E. Ukkonen, "On-Line Construction of Suffix Trees," Algorithmica, vol. 14, no. 3, pp. 249-260, 1995.
[15]. Dekang Lin, "An Information-Theoretic Definition of Similarity", Proc. of 15th International Conference Conference on Machine Learning, pp. 296-304, Wisconsin, USA, 1998.