Text Extraction for an Agglutinative Language

Sankar K, Vijay Sundar Ram R and Sobha Lalitha Devi
AU-KBC Research Centre
MIT Campus of Anna University
Chennai, India

***Abstract*- The paper proposes an efficient algorithm for sentence ranking based on a graph theoretic ranking model applied to text summarization task. Our approach employs word frequency statistics and a word positional and string pattern based weight calculation for weighing the sentence and to rank the sentences. Here we have worked for a highly agglutinative and morphologically rich language, Tamil.**

I. INTRODUCTION

The enormous and on-going increase of digital data in internet, pressurize the NLP community to come up with a highly efficient automated text summarization tools. The research on text summarization is boosted by the various shared tasks such as TIPSTER SUMMAC Text Summarization Evaluation task, Document Understanding conference (DUC 2001 to 2007) and Text Analysis conferences.

A variety of automated summarization schemes have been proposed recently. NeATS [4] is a sentence position, term frequency, topic signature and term clustering based approach and MEAD [10] is a centroid based approach. Iterative graph based Ranking algorithms, such as Kleinberg's HITS algorithm [3] and Google's PageRank [1] have been successfully used in web-link analysis, social networks and more recently in text processing applications [8], [7], [2] and [9]. These iterative approaches have a high time complexity and are practically slow in dynamic summarization. The works done in Text Extraction for Indian languages is comparatively less.

In this paper we have discussed a novel automatic and unsupervised graph based ranking algorithm, which gives improved results compared to other ranking algorithms in the context of the text summarization task. Here we have worked for Tamil.

II. TEXT SUMMARIZATION AND TEXT RANKING

Text summarization is process of distilling the most important information from the set of source to provide a abridge version for particular user and tasks. The text summarization is also done by ranking in the sentences in the given source test. Here we have proposed a graph based text ranking approach.

Graph based algorithm is essentially a way of deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. The basic idea here is that of 'voting' or 'recommendation'. When one vertex links to the other vertex, it is like casting a vote for that vertex. The vertex becomes important when it links with more number of vertices. The importance of vertex casting the vote determines how important the vote itself is [10].

The proposed graph based text ranking algorithm consists of two types of measure (1) Word Frequency Analysis; (2) A word positional and string pattern based weight calculation. Based on the above two scores, the ranking of sentences is done.

The algorithm is carried out in two phases. The weight metric obtained at the end of each phase is averaged to obtain the final weight metric. Sentences are sorted in descending order of weight.

A. Graph

Let G (*V*, *E*) be a weighted undirected complete graph, where *V* is set of vertices and *E* is set of weighted edges.

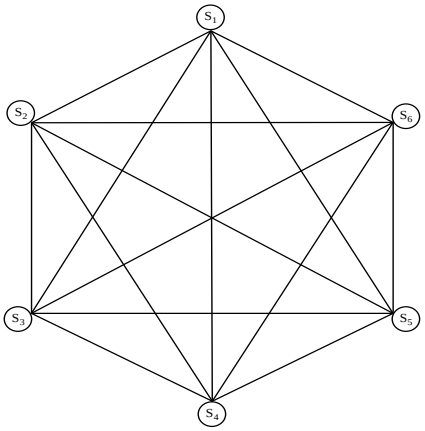

Fig. 1. A complete undirected graph

In figure 1, the vertices in graph G represent the set of all sentences in the given document. Each sentence in G is related to every other sentence through the set of weighted edges in the complete graph.

B. Phase 1 : Word Frequency Analysis

In Word Frequency Analysis, we find the affinity weight (AW) for each word in the sentence by using the formula 1. The sentence weight (SW) is calculated by averaging the AW of all words in the sentence.

The affinity weight for each word is calculated by frequency of the given word in the sentence divided by number of words in the sentence.

Word Frequency in Tamil:

As Tamil is a morphologically rich and a highly agglutinative language, getting the frequency of the words is not straight forward. The text has to be preprocessed with a morph-analyser to collect the corresponding root words, as all the words in the sentences will be in inflected form (root + suffixes). Given a word to the morph-analyser, it will split the word into root and its suffix and return the valid root word alone. Example

மரங்களை -> மரம் + கள் + ஐ -> மரம்

marangkaLai -> maram + kaL + ai -> maram

(tree + plural +acc) tree plural acc tree

Let the set of all sentences in document $S = \{s_i \mid 1 \le i \le n\}$, where n is the number of sentences in S. For a sentence $s_i = \{w_j \mid 1 \le j \le m_i\}$ where m_i is the number of words in sentence s_i, $(1 \le i \le n)$ the affinity weight AW of a word w_j is calculated as follows:

$$AW(w_j) = \frac{\sum_{\forall w_k \in S} IsEqual(w_j, w_k)}{WC(S)} \qquad (1)$$

where S is the set of all sentences in the given document, w_k is a word in S, WC (S) is the total number of words in S and function $IsEqual(x, y)$ returns an integer count 1 if x and y are equal else integer count 0 is returned by the function.

Then, we find the sentence weight SW (s_i) for each sentence s_i $(1 \le i \le n)$ as follows:

$$SW(s_i) = \frac{1}{m_i} \sum_{\forall w_j \in s_i} AW(w_j) \qquad (2)$$

At the end of phase 1, the graph vertices hold the sentence weight as shown in figure 3 for graph constructed using the following sentences.

[1] தாஜ் மகால், இந்தியாவிலுள்ள நினைவுச்சின்னங்களுள், உலக அளவில் பலருக்குத் தெரிந்த ஒன்றாகும்.

Taj Mahal, among the memorials in India, is known word wide.

[2] இது ஆக்ராவில் அமைந்துள்ளது.

This is located in Agra.

[3] முழுவதும் பளிங்குக் கற்களாலான இக்கட்டிடம், ஆக்ரா நகரில் யமுனை ஆற்றின் கரையில் கட்டப்பட்டுள்ளது.

This building fully made of marbles is built on the shores Yamuna river in Agra.

[4] இது காதலின் சின்னமாக உலகப் புகழ் பெற்றது.

This is world famous as a symbol of love.

[5] ஏழு உலக அதிசயங்களின் புதிய பட்டியலில் தாஜ் மகாலும் சேர்க்கப்பட்டுள்ளது.

In the new seven wonders of the world Taj Mahal is also included.

[6] இக்கட்டிடம் முகலாய மன்னனான ஷாஜகானால், இறந்து போன அவனது இளம் மனைவி மும்தாஜ் நினைவாக 22,000 பணியாட்களைக் கொண்டு 1631 முதல் 1654 ஆம் ஆண்டுக்கு இடையில் கட்டிமுடிக்கப்பட்டது.

Mughal emperor Sharjahan built this building using 22,000 workers, from 1631 to mid of 1654, in memory of young wife Mumthaz .

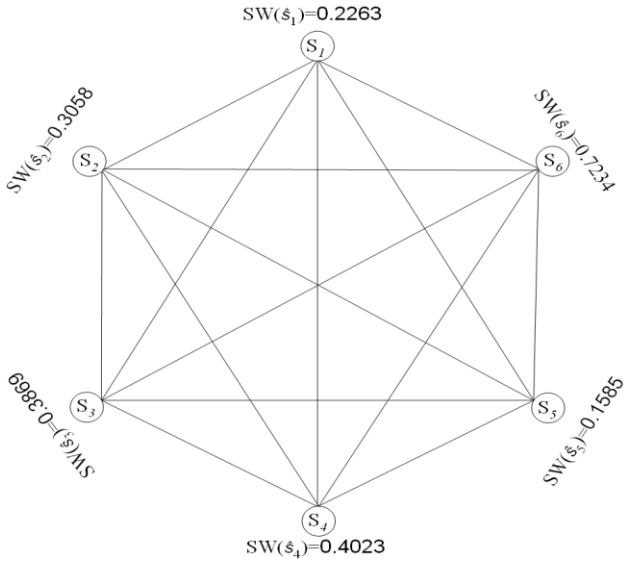

Fig. 3. Sample graph of Sentence weight calculation in phase 1.

C. Phase 2 : A Word Positional and String Pattern Based Weight Calculation

In phase 2, a word positional and string pattern based weight in all the vertices is calculated using Levenshtein Similarity measure (LSW), which uses Levenshtein Distance for calculating the weight.

The vertex weight is calculated by summing all the LSW and dividing it with number of sentences.

Levenshtein Distance

Levenshtein distance (LD) is a measure of the similarity between two strings source (s) and target (t). The distance is the minimum number of deletions, insertions, or substitutions required to transform s into t.

The *LD* algorithm is illustrated by the following example

LD (RAIL, MAIL) is 1

LD (WATER,METER) is 2

Similarly, the LD calculation is same for words in Tamil, but there are three and two character letters in Tamil which we have to consider as single character while calculating the distance, as shown below.

LD(சொல்,வால்) is 1

LD(வரும்படி,என்னப்படி) is 4

Levenshtein Similarity Weight

Levenshtein Similarity Weight is calculated between the sentences, considering two sentences at an instance. This is calculated by dividing the difference of maximum length between two sentences and LD between the two sentences by maximum length between two sentences as shown in formula 6.

Consider two sentences, *sentence*1 and *sentence*2 where ls_1 is the length of *sentence*1 and ls_2 be the length of *sentence*2. Compute *MaxLen*=maximum (ls_1, ls_2). Then *LSW* between *sentence*1 and *sentence*2 is the difference between *MaxLen* and *LD*, divided by *MaxLen*. Clearly, *LSW* lies in the interval 0 to 1. In case of a perfect match between two words, its *LSW* is 1 and in case of a total mismatch, its *LSW* is 0. In all other cases, 0 < *LSW* <1. The *LSW* metric is illustrated by the following example.

Considering these strings as sentences,

LSW (ABC, ABC) =1

LSW (ABC, XYZ) =0

LSW (ABCD, EFD) =0.25

Similarly

LSW (என்னப்படி, வருப்படி) = (6-4)/6 = 0.3334

Levenshtein similarity weight is calculated by the equation

$$LSW(s_i, s_j) = \frac{MaxLen(s_i, s_j) - LD(s_i, s_j)}{MaxLen(\hat{s}_i, s_j)} \qquad (6)$$

where, s_i and s_j are the sentences.

Hence before finding the LSW, we have to calculate the LD between each sentence.

Let $S = \{s_i \mid 1 \le i \le n\}$ be the set of all sentences in the given document; where n is the number of sentences in S. Further, $s_i = \{w_j \mid 1 \le j \le m\}$, where m is the number of words in sentence s_i.

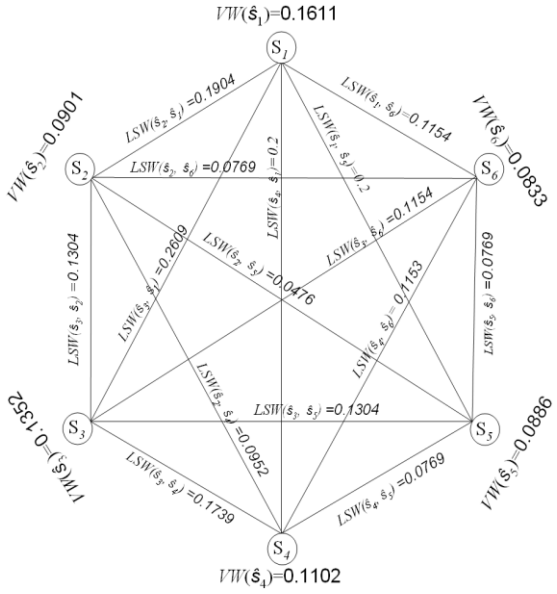

Fig. 4. Sample graph for Sentence weight calculation in phase 2

Each sentence s_i ; $1 \le i \le n$ is represented as the vertex of the complete graph as in figure 4 and $S=\{s_i|1 \le i \le n\}$. For the graph in figure 4, find the Levenshtein similarity weight *LSW* between every vertex using equation 6. Find vertex weight (*VW*) for each string s_i ; $1 \le l \le n$ by

$$VW(s_l) = \frac{1}{n} \sum_{\forall s_i \ne s_l \in S} LSW(s_l, s_i) \qquad (7)$$

3. Text Ranking

Obtaining the sentence weight ($SW(s_i)$) and the vertex weight $VW(s_i)$, the ranking score is calculated is the formula 8, where the average of the two scores are found.

The rank of sentence s_i; $1 \le i \le n$ is computed as

$$Rank(s_i) = \frac{SW(s_i) + VW(\hat{s}_i)}{2}; 1 \le i \le n \qquad (8)$$

where, $SW(s_i)$ is calculated by equation 2 of phase 1 and $VW(\hat{s}_i)$ is found using equation 7 of phase 2. The ranking scores for the sentences (s_i; $1 \le i \le n$,) are arranged in descending order of their ranks.

$SW(s_i)$ in phase 1 holds the sentence affinity in terms of word frequency and is used to determine the significance of the sentence in the overall raking scheme. $VW(\hat{s}_i)$ in phase 2 helps in the overall ranking by determining largest common subsequences and other smaller subsequences then assigning weights to it using *LSW*. Further, since named entities are represented as strings, repeated occurrences are weighed efficiently by *LSW*, thereby giving it a relevant ranking position.

4. Evaluation And Discussion

We have used the ROUGE evaluation toolkit to evaluate the proposed algorithm. ROUGE, an automated summarization evaluation package based on N-gram statistics, is found to be highly correlated with human evaluations [4].

The evaluations are reported in ROUGE-1 metrics, which seeks unigram matches between the generated and the reference summaries. The ROUGE-1 metric is found to have high correlation with human judgments at a 95% confidence level, so this is used for evaluation. The present Graph-based Ranking Algorithms for Text Extraction works with Rouge score of 0.4723.

We manually created the reference summaries for 150 documents taken from online news articles. The reference summaries and the summaries obtained by our algorithm are compared using the ROUGE evaluation toolkit, which is presented in Table 1. For each article, our proposed algorithm generates a 100-words summary.

TABLE I
Rouge Score

	Score
ROUGE-1	0.4723

The methodology performs well even for the agglutinative languages. For the word frequency calculation we feed only the root words instead of the agglutinative words to get proper frequency count. In the phase 2 where the Levenshtein Similarity Weight, the distance varies more as the all the sentences have different inflected and agglutinative words. Again in word frequency, the pronouns occurring in the same sentence, which actual reference to one of the noun phrase (occurs instead of a noun phrase), cannot to be counted.

Conclusions

In this paper, we introduced Graph Based Ranking algorithm for text ranking. Here we have worked for Tamil, a south Dravidian language. Here we have shown the necessity of getting the root words for Text ranking. The architecture of the algorithm helps the ranking process to be done in a time efficient way. This text ranking algorithm is not a domain specific and also does not require any annotated corpora. This approach succeeds in grabbing the most important sentences based on the information exclusively from the text itself; whereas other supervised ranking systems do this process by training on summary collection.

5. CONCLUSIONS

In this paper, we introduced Graph Based Ranking algorithm for text ranking. Here we have worked for Tamil, a south Dravidian language. Here we have shown the necessity of getting the root words for Text ranking. The architecture of the algorithm helps the ranking process to be done in a time efficient way. This text ranking algorithm is not a domain specific and also does not require any annotated corpora. This approach succeeds in grabbing the most important sentences based on the information exclusively from the text itself; whereas other supervised ranking systems do this process by training on summary collection.

REFERENCES

[1] Brin and L. Page. 1998. The anatomy of a large-scale hypertextualWeb search engine. *Computer Networks and ISDN Systems*, 30 (1 – 7).

[2] Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, July.

[3] Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604-632.

[4] Lin and E.H. Hovy. From Single to Multi-document Summarization: A Prototype System and its Evaluation. *In Proceedings of ACL-2002*.

[5] Lin and E.H. Hovy. 2003a. Automatic evaluation of summaries using n-gram co-occurrence statistics. *In Proceedings of Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, May.

[6] Lin and E.H. Hovy. 2003b. The potential and limitations of sentence extraction for summarization. *In Proceedings of the HLT/NAACL Workshop on Automatic Summarization*, Edmonton, Canada, May.

[7] Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004) (companion volume)*, Barcelona, Spain.

[8] Mihalcea and P. Tarau. 2004. TextRank - bringing order into texts. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain.

[9] Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. *In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

[10] Radev, H. Y. Jing, M. Stys and D. Tam. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919-938, 2004.