Query Specific ROCK Clustering Algorithm for Text Summarization

Ms. Laxmi S. Patil[1], Prof. M. S. Bewoor[2] and Dr. S. H. Patil[3]
Department of Computer Engineering, Bharati Vidyapeeth Deemed University
College of Engineering, Pune, India.
[1](M.Tech Student)
[2] (Asst. Professor,)
[3](Head, Computer Engineering Department.)

ABSTRACT

The idea of Data Mining has become very popular in recent years. Data Mining is the notion of all methods and techniques, which allow analyzing very large data sets to extract and discover previously unknown structures and relations out of such huge heaps of details. Data clustering is an important technique for exploratory data analysis. Clustering is a data mining (machine learning) technique used to place data elements into related groups without advance knowledge of the group definitions. Clustering is nothing but Grouping of objects into different sets, or the partitioning of a data set into subsets (clusters), In this paper we provides in depth explanation of implementation adopted for ROCK (RObust Clustering using linKs) clustering algorithm. We propose a novel concept of links to measure the similarity/proximity between a pair of data points. We develop a robust hierarchical clustering algorithm ROCK that employs links and not distances when merging clusters.

Keywords - **Document graph, NLP- Natural Language Processing, Query-specific document summary, ROCK(RObust Clustering using linKs).**

I. INTRODUCTION

1. Overview

Over time, there have been various approaches to automatic text summarization. One of the techniques of document summarization is summarization through various clustering algorithms. Again these clustering algorithms are classified in two categories as a) Query-Based Text Summarization System uses standard retrieval methods to map a query against a document collection and to create a summary & b) The second system, the Concept-Based Text Summarization System, creates a query-independent document summer [2]. The ROCK algorithm which we are going to implement is of first type i.e. it creates query-specific document summary. The main aim of this research work is to combine both approaches of document clustering and query dependent summarization with natural language processing (NLP) based. This mainly includes applying different clustering algorithms on a text document. Create a weighted document graph of the resulting graph based on the keywords. And obtain the optimal tree to get the summary of the document [4]. Fig. 1 shows the architecture diagram of the system. As shown in figure there are four main blocks: a block for uploading and processing text file and making document graph, a block for clustering and making clustered graph, a block for making weighted clustered document graph, the last block for generating summary for fired query.

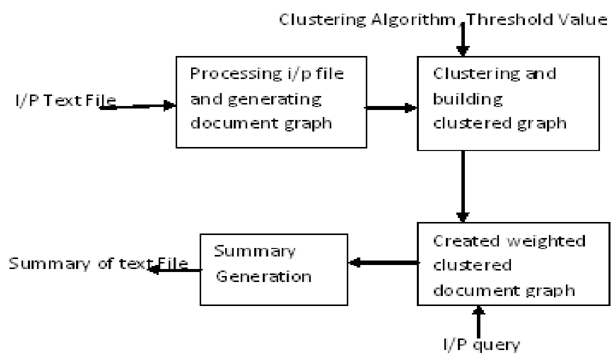
Clustering Algorithm, Threshold Value
I/P Text File
Processing i/p file and generating document graph
Clustering and building clustered graph
Summary of text File
Summary Generation
Created weighted clustered document graph
I/P query

Fig 1 Architecture Diagram

Block 1: Processing input file and generating document graph:
This block is needed to accept the text file only. It is responsible to upload text file, to process the file i.e. to form nodes for every newline contents. It is also responsible for generating weight from each node to very other node

Block 2. Clustering node and building clustered graph: This block is responsible for choosing a clustering algorithm out of two. It also accepts the threshold, so that can check the similarity between the clusters up to that level. It is responsible for making clusters. [6,7]

Block 3. Creating weighted document clustered graph: This block is responsible to accept the fired query. It is responsible to check the similarities between the query a contents and the contents in the clusters. It then build weighted clustered document graph.

Block 4. Summary generation: This block is responsible for generating the summary of the clusters we formed, as a response for fired query. It generated the minimal clusters and after finding the weight of the node for fired query, it gives top most summaries.

2. Goal
The goal is to use ROCK clustering algorithm for query dependent clustering of nodes in text document, and finding query dependent summary. The summary will be compared and best algorithm will be suggested for query dependent clustering using different clustering techniques. This technique will help end users to prepare query dependent summary of text documents in which they are interested.
-The proposed work will be mainly focused on summarization of text files (i.e. .txt) with NLP.

-The proposed work will be limited to clustering of text files of standard files related to the topic popular amongst researchers will be used.

-Only ROCK clustering algorithm methods are considered for generating cluster based graph.

II. SYSTEM IMPLENETATION

Implementation is very important phase; the most critical stage in achieving a successful new system so that the new system will work is effective. After the system is implemented the testing can be done. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system. [5, 6, 7].

The total workflow is divided into following modules:

Module 1: Processing the input text file and creating the document graph.
The system accepts input text file. The file is read and stored into a string. The string is then split by the newline keyword. The split file is assigned to the string array as the split function returns the string array. The array contains paragraphs which are further treated as nodes.
The next stage is to find the similarity between the nodes that means finding the similarity edges between nodes and finding their similarity or weight. Each paragraph becomes a node in the document graph.
The document graph G (V, E) of a document d is defined as follows:
□ □*d* is split to a set of non-overlapping nodes *t (v), v* □ □*V.*
□ □An edge *e (u, v)*□*E* is added between nodes *u, v* □ □*V* if there is an association between *t (u)* and *t (v)* in d. Hence, we can view G as an equivalent representation of d, where the associations between text fragments of d are depicted.

Module 2: Adding Weighted Edges to Document Graph (Note: Adding weighted edge is query independent)
A weighted edge is added to the document graph between two nodes if they either correspond to adjacent node or if they are semantically related, and the weight of an edge denotes the degree of the relationship. Here two nodes are considered to be related if they share common words (not stop words) and the degree of relationship is calculated by "Semantic parsing". Also notice that the edge weights are query-independent, so they can be pre-computed. [2,8,9]
The following input parameters are required at the pre computation stage to create the document graph:

1. Threshold for edge weights: Only edges with weight not below threshold will be created in the Adding weighted edge is the next step after generating document graph. Here for each pair of nodes *u, v* we compute the association degree between them, that is, the score (weight) EScore (e) of the edge e (u, v). If Score (e) ≥ threshold, then *e* is added to E. The score of edge e (u, v) where nodes u, v have text fragments t(u), t(v) respectively is:

$$\text{EScore} = \frac{\sum ((tf(t(u),w + tf(t(v),w)).idf(w)}{\text{size}(t(u)) + \text{size}(t(v))}$$

Where t f (d, w) is the number of occurrences of w in d,
id f (w) is the inverse of the number of documents containing w, and
size(d) is the size of the document (in words).That is, for every word w appearing in both text fragments we add a quantity equal to the tf□ idf score of w. Notice that stop words are ignored.
If EScore >= Threshold, the edge is added to the document graph.
The graph is stored into tabular form as shown below

First_Node	Second_Node	Edge Weight
1	2	0.5
1	3	0.8
. .	. .	. .
15	16	0.6
15	17	0.7

Table 1. Nodes and Node weights

Module 3: Document Clustering
Clustering is grouping of similar nodes (The nodes which shows degree of closure greater than or equal to the Cluster Threshold specified by the user) into a group. The clustering algorithm used is ROCK (RObust Clustering using LinKs).
Algorithm for Rock:
The steps involved in clustering using ROCK are described in Fig 2. After drawing a random sample from the database, a hierarchical clustering algorithm that employs links is applied to the sampled points. Finally, the clusters involving only the sampled points are used to assign the remaining data points on disk to the appropriate clusters. In the following subsections, we first describe the steps performed by ROCK in greater detail

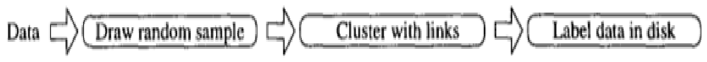

Fig. 2 Steps performed by ROCK

Algorithm:
1. Obtain random sample of data

2. Cluster data using link based agglomerative technique. Use a 'goodness' measure to determine which points are merged
3. Using these clusters, merge with remaining data.

Fig. 3 represents ROCK`s hierarchical clustering algorithm. It accepts as input the set S of N sampled points to be clustered (that are drawn randomly from the original data set), and the number of desired clusters k. The procedure begins by computing the number of links between pairs of points in Step 1. Initially, each point is separate cluster. For each cluster i, we build a local heap q[i] and maintain the heap during the execution of the algorithm. q[i] contains every cluster j such that link[i,j] is non-zero. The clusters j in q[i] are ordered in the decreasing order of the goodness measure with respect to i, g(i,j).

In addition to the local heaps q[i] for each cluster i, the algorithm also maintains an additional global heap Q that contains all the clusters. Furthermore, the clusters in Q are ordered in the decreasing order of their best goodness measures. Thus, g(j,max(q[j])) is used to order the various clusters j in Q, where max(q[j]), the max element in q[j], is the best cluster to merge with cluster j. At each step, the max cluster j in Q and the max cluster q[j] are the best pair of clusters to be merged

```
procedure cluster(S, k)
begin

    1.  link := compute links(S)
    2.  for each s ∈ S do
    3.      q[s] := build local heap(link, s)
    4.  Q := build global heap(S, q)
    5.  while size(Q) > k do
    6.  {
    7.          u := extract max(Q)
    8.          v := max(q[u])
    9.          delete(Q, v)
    10.         w := merge(u, v)
    11. for each x ∈ q[u] ∪ q[v] do
    12.   {
    13.     link[x, w] := link[x, u] + link[x, v]
    14.     delete(q[x], u); delete(q[x], v)
    15.     insert(q[x], w, g(x, w)); insert(q[w], x, g(x, w))
    16.     update(Q, x, q[x])
    17.   }
    18. insert(Q, w, q[w])
    19. deallocate(q[u]); deallocate(q[v])
    20.  }
end
```

Fig. 3 Rock's hierarchical clustering algorithm

For every point, after computing a list of its neighbours, the algorithm considers all pairs of its neighbours. For each pair, the point contributes one link. If the process is repeated for every point and the link count is incremented for each pair of neighbours, then at the end, the link counts for all pairs of points will be tained. If Mi is the size of the neighbour list for point i, then for point i, we have to increase the link count by one in M^2i entries. This, the complexity of the algorithm is the sum of M^2i which is O(N * Mm * Ma), where Ma and Mm are the average and maximum number of the neighbours for a point, respectively. In the worst case, the value of Mm can be n in which case the complexity of the algorithm becomes O(Ma * N^2). In practice, we expect Mm to be reasonably close to Ma and thus, for these cases, the complexity of the algorithm reduces to O(M^2a * n) on average.

```
procedure compute links(S)
begin
    1. Compute nbrlist[i] for every point i in S
    2. Set link[i, j] to be zero for all i, j
    3. for i := 1 to n do
    4. {
    5.     N := nbrlist[i]
    6.     for j := 1 to |N| − 1 do
    7.         for l := j + 1 to |N| do
    8.             link[N[j], N[l]] := link[N[j], N[l]] + 1
    9.     }
end
```

Fig. 4 Computation of Links

III. SUGGESTED FUTURE WORK

In this system ROCK clustering algorithms is used for context sensitive text summarization. Likewise another algorithm i.e. Graph Theoretic clustering algorithms can be considered for clustering the nodes in text file and by comparing the query dependent summary using certain criteria, the best clustering algorithm can be suggested.
Furthermore same technique can be applied on different file formats and best clustering algorithm can be suggested for different file formats.

IV. CONCLUSION

In this work we presented a structure-based technique to create query-specific summaries for text documents. In particular, we first create the document graph of a document to represent the hidden semantic structure of the document and then perform keyword proximity search on this graph. We show with a user survey that our approach performs better than other state of the art approaches. Furthermore, we show the feasibility of our approach with a performance evaluation.

REFERENCES

Journal Papers:

[1] M Ramakrishna Varadarajan, Vangelis Hristidis , *"A System for Query-Specific Document SummarizatioSn"*

[2] Mahmoud El-Haj, Udo Kruschwitz, Chris Fox, "*Experimenting with Automatic Text Summarization for Arabic*"

[3] Vishal Gupta, "*A Survey of Text Summarization Extractive Techniques*"

[4] Mohamed Abdel Fattah, and Fuji Ren, "*Automatic Text Summarization*"

[5] Parul Agarwal, M. Afshar Alam, anjit Biswas, "*Analyzing the agglomerative hierarchical Clustering Algorithm for Categorical Attributes*"

[6] Jie Tang, Limin Yao, and Dewei Chen , "*Multi-topic based Query-oriented Summarization*"

[7] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Proceedings of the ACL-04 Workshop: Text Summarization Branches Out, pages 74–81, Barcelona, Spain.

[8] Luhn H. P. 1958, The automatic creation of literature abstracts, IBM Journal, pages 159-165

Books:

[9] The complete Reference of .NET-by Matthew, Tata MacGraw Hill Publication Edition