S-SUM: A System for Summarizing the Summaries

C Ravindranath Chowdary
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU),
Varanasi, India 221 005
rchowdary.cse@iitbhu.ac.in

P Sreenivasa Kumar
Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
Chennai, India 630 036
psk@iitm.ac.in

ABSTRACT

Query-specific summarization of multiple documents is a useful task in the current day context of the WWW, that is containing huge amount of information. When different summarizers have access to different sets of documents for a query, generating a summary of the summaries produced by the multiple summarizers becomes an interesting and useful task. In this paper, we propose an efficient solution for this problem. Sentences from the individual summaries are used to construct an Integrated Linear Structure (ILS) and are given unique position numbers. All the sentences in the ILS are then assigned weights that reflect the importance of the sentences to the given query. Sentences are selected according to these weights using the Maximal Marginal Relevance (MMR) approach for inclusion into the final summary of summaries. Finally, the sentences in the final summary are sorted based on their position numbers given using ILS. Experimental results show that *S-SUM* is efficient.

1. INTRODUCTION

Huge amount of information is present in the World Wide Web and a large amount of information is being added to the WEB regularly. Information on a topic is distributed across the multiple pages/documents. It is a nontrivial task for a user to go through all these documents to find the information of her interest. Most of the times, there will be a lot of redundancy in the information content and it will be a tedious task for the user to read all the documents. To overcome this problem, query specific multi-document summarizers were proposed [13, 15, 19, 20, 10]. With the increasing need for quality summarizers, this field is gaining momentum.

In extractive summary generation, the sentences are selected from the documents and are arranged in a meaningful order. Summaries can be generated either from a single document or from multiple documents and a summary can be either generic [18, 5] or query-specific. In this paper, we deal with query-specific extractive summarization. Choosing a sentence for inclusion in a summary is determined by the amount of importance it has with the query. The challenge lies in assigning importance to a sentence. Sentences in all the documents should be given scores based on the importance they have with the query posed by the user.

The approach used by a typical query specific multiple document summarizer is: Sentences from *all* the documents are arranged as nodes in a graph. Similarities among all the sentences are calculated using a *similarity* measure. Sentence s is selected into the summary based on both the importance of s with the query and the importance of sentences that have high similarity with s. The quality of summaries generated by these systems is good but these systems cannot be used for on-line/real time purpose due to their high computational complexity. The complexity is directly proportional to the number of sentences/nodes in the graph. One possible solution is to divide a huge set of documents into smaller sets and generate summaries on these smaller sets. By doing so, efficiency would be increased but there may be some information loss. Further, these individual summaries are to be summarized as the final summary has a restriction on its size *i.e.*, 250 words, 500 words, *etc.*

1.1 Motivation

A search engine retrieves the ranked list of documents for a given query. Each search engine may retrieve a different set of documents. These individual documents retrieved by search engines can be summarized independently *i.e.*, a different summarizer could be used for each of the search engine. In this scenario, if a user wants the overall gist/summary on a topic, the only solution is to go through all the summaries generated by different summarizers. Also, it is likely to have a fair amount of overlap between the retrieved sets of documents. This overlap of information in documents may lead to overlap of information in summaries. Therefore the individual summaries may have both diverse and redundant information. If there is a system to generate a summary from the summaries generated by different summarizers, then it would save a lot of users time.

Another scenario is if n users write their brief opinion on a topic, it would be of great help to generate a summary of these n opinions. This motivated us to develop a system that summarizes opinions/summaries. In this paper, we propose and report experimental results of a system, called *S-SUM*, that generates a *summary* from the summaries/opinions.

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

1.2 Introduction to S-SUM

To the best of our knowledge, there are no extractive summarizers in the literature that take query-specific summaries as input. An initial solution to this problem can be applying a query-specific multiple document summarizer on the given summaries. However, there are issues to be considered while generating summary of summaries. They are 1) ordering of sentences in the final summary. Since sentences are selected from different summaries, it is a challenging task to have a logical flow in the final summary. 2) All the summaries are query specific and the sentences within the summary are highly related to the query. So, direct centrality [15, 13] based sentence score may not give good results. Both these challenges are effectively addressed in this paper. In this paper, sentences are given scores as a combination of both centroid [5] and centrality [15, 13] methods.

2. RELATED WORK

Text summarization has picked up pace in the recent years. Multi-document generic summary generation is discussed in MEAD [5]. MEAD generates summary by following centroid based approach. Given a set of documents about a particular topic *i.e.*, a cluster of documents, a centroid of the cluster is calculated. Each sentence in the cluster is given a score- with respect to the centroid obtained, similarity with the first sentence in the document and the relative position with respect to the first sentence. Sentences are selected in the decreasing order of sentence scores and are arranged chronologically.

Centrality based summarization approaches are discussed in [1, 8, 9, 7, 17]. Degree centrality is discussed in [1] and eigenvector centrality is discussed in [8, 9, 7]. Degree centrality of a node is calculated by counting the number of other nodes it is connected to i.e., degree of a node. An edge is placed between two nodes if and only if there is a considerable amount of overlap of words between the nodes. Concept of bushy path was introduced in [1]. A node with high degree is called as bushy node. A path connecting top n bushy nodes is bushy path. Eigenvector centrality of a node is calculated by taking into consideration both the degree of the node and the degree of the nodes connecting to it, this is inspired by PageRank[2].

In [12] topic focused single document summarization is addressed. Document is modeled as a graph. Irrespective of the threshold condition, an edge is placed between adjacent nodes in the graph. Node score is calculated with respect to the query. A minimal set of nodes are picked which will cover(include) all the query terms. This minimal set may not be having direct edges among them, so, intermediate nodes are added to make the selected set of nodes a connected sub graph of the original graph.

Query specific summary generation is discussed in [11]. In [11] a document is modeled as a graph and similarity between nodes is calculated using *cosine similarity* measure. An edge is present between nodes if the similarity value exceeds a threshold. Node scores are calculated by taking both term frequencies(tf) and inverse document frequencies (idf) into consideration. A node gets high score if it is connected to the nodes with high score. The node scores are computed iteratively till the values converge. Query is considered as a node of the document graph in[13] and pairwise similarity between sentences is calculated and an edge is placed between nodes if the similarity value is greater than *zero*. As the query is part of the graph, centrality based approach is followed to select the nodes into the summary.

Generating Non-Redundant summaries is addressed in [3]. Node scores are calculated based on the similarity w.r.t the query and the summary is generated incrementally. To start with, a node with highest score is selected into the summary. All the scores of remaining nodes are recalculated based on both their current node scores and the similarity with the nodes already selected into the summary. From the recalculated scores, the node with the highest score will be added to the summary. In most of the models discussed above, node scores are calculated by following ideas similar to PageRank[2] and HITS[4] and edge scores are calculated based on the amount of similarity between nodes. All the models mentioned above address the issue of generating summary from a single document or multiple documents for a generic/specific purpose. None of these systems address the problem of generating a summary of summaries that are generated from different summarizers on different sets of documents for a given topic.

3. FRAMEWORK

The task in query specific summary generation is to extract sentences from the documents. These sentences should be very relevant to the query. A summary is said to be *complete* if it contains information about the whole query. Completeness is calculated based on the presence of query terms in a summary. A summarizer will extract a sentence from the documents based on the score given to it. The order in which the sentences are extracted need not follow a logical sequence. Hence, these sentences have to be rearranged to get a logical flow to the summary. The quality of having logical flow in a summary is termed as *coherence*. In multi-document summarization, some amount of information would be present in more than one document. If the information that is repeated across the multiple documents is found to be important, such information should be present in the summary but should not be repeated. Such non-repetition of information in a summary is termed as *non-redundancy*. In this paper, coherence and non-redundancy are addressed explicitly and completeness is achieved implicitly.

In this paper, we model all the sentences from all the summaries as nodes in a graph. Each sentence is considered as a vector of words. An edge is placed between two sentences if similarity between them is above a threshold. *Similarity* is calculated as given in Equation 1

$$sim(\overrightarrow{n_i}, \overrightarrow{n_j}) = \frac{\overrightarrow{n_i}.\overrightarrow{n_j}}{|\overrightarrow{n_i}||\overrightarrow{n_j}|} \quad (1)$$

where $\overrightarrow{n_i}$ and $\overrightarrow{n_j}$ are term vectors for the nodes n_i and n_j respectively. Weight of each term(t) in the term vector is calculated as $tf_t * isf_t$ where tf_t is the *term frequency* and isf_t is *inverse sentential frequency*. isf_t is calculated as $log(\frac{n}{n_t+1})$ where n is the total number of nodes in the graph and n_t is the number of nodes containing the term t in the graph. All the stop words are removed and the remaining words are stemmed before computing the weights.

4. INTEGRATED LINEAR STRUCTURE (ILS)

In this section, we discuss the construction of *ILS*. *ILS* contains all the sentences of the summaries. Each sentence in *ILS* is assigned a position number. Sentences of the final summary are arranged in ascending order of these position numbers. We observed that this arrangement of sentences ensures coherence in the final summary. In this paper, we assume that the individual summaries from which *ILS* is constructed are coherent. We also observed that a summary generated on a smaller set of documents will be more coherent than a summary generated on a larger set. The summary with the highest number of sentences is taken as the base summary. Each sentence in the base summary is taken as a *context* node. Position numbers are assigned to these context nodes as the integral multiples of the parameter "gap". *ILS* is constructed by inserting every sentence from the remaining summaries into the base summary. Each non-context sentence will be in the *Neighbourhood* of a context node. *Neighbourhood* of a context node c is the set of all sentences from remaining summaries that have highest similarity with c when compared other context nodes. The similarity value should be above a threshold. This threshold has to be chosen appropriately. If the similarity value of a node is less than this threshold with all the context nodes, then the node is introduced as a new context node.

For example, if *gap* is taken as 200, the position values of the contexts will be *0, 200, 400, 600, 800, etc.* All the nodes that are neighbours to the context node at 200 will be inserted between the context nodes with position numbers 200 and 400 respectively *i.e.,* all the neighbours of 200 will have position values in the range [201,399]. The exact procedure for positioning of neighbours is described in the later part of this section. The concept of *neighbourhood* is introduced based on the following observations *i,e.,*

- As all the summaries are generated for a query/topic, it is highly likely to have similarity between sentences across the summaries.
- These similar sentences may be having information on a topic.
- These similar sentences may have repetition of information.

So, all the nodes that have similar information are introduced as neighbours to their context node. This arrangement is useful to achieve logical flow in the final summary. Usually, the nodes with new information are introduced as new context nodes. A node is said to have a new information if it is not similar with any of the existing context nodes. In this case, it is better to introduce it as a new context node. This will be useful to identify the theme of the summary, discussion of which is out of scope of this paper.

In Algorithm 1, the construction of ILS is given. The base summary, S_0, is identified and position numbers are assigned to the nodes in the base summary. Each node in the base summary is made as a context node. The parameter *gap* is a constant that is chosen appropriately to accommodate the neighbours. In Lines 13 - 29, similarity of a sentence d with all the context nodes is computed. The context node with which d has highest similarity is found. If this similarity value exceeds a threshold, $\eta(>0.1)$, then d is assigned a position number as explained in Algorithm 2. Otherwise, d is made as a new context node as outlined in Algorithm 3.

Algorithm 1 Construction of Integrated Linear Structure

```
1: Input: Summaries arranged in the decreasing order of
   their size(number of sentences)
2: Output: Integrated Linear Structure ILS
3: Integrated Linear Structure ILS = S_0 {//base sum-
   mary}
4: k = |S_0|
5: j = 0
6: while j < k do
7:    {//Assign position to each node n_j in ILS}
8:    position(n_j) = j * gap
9:    j++
10: end while
11: i = 1
12: while i ≤ number of Summaries do
13:    for each node d ∈ S_i do
14:       {// find the context node,
          MaxSimilarContextNode, with which d has
          maximum similarity}
15:       for each context node n_j do
16:          if MaxSimilarContextNode = Null then
17:             MaxSimilarContextNode = n_j
18:          else
19:             if Sim(MaxSimilarContextNode, d) <
                Sim(n_j, d) then
20:                MaxSimilarContextNode = n_j
21:             end if
22:          end if
23:       end for
24:       if sim(d, MaxSimilarContextNode) ≥ η then
25:          Introduce d as a neighbour in the context of
             MaxSimilarContextNode as explained in Algo
             2
26:       else
27:          Make d as the new context in ILS as explained
             in Algo 3
28:       end if
29:    end for
30:    i++
31: end while
```

In Algorithm 2, inserting a node d in the neighbourhood of a context node is discussed. All the neighbours of a context will have increasing position values in the decreasing order of their similarity with the context. For example, if the context node at *200* has nodes with position numbers *201, 202 and 203* as neighbours then *sim(nodeAtPosition(200), nodeAtPosition(201))* will be greater than *sim(nodeAtPosition(200), nodeAtPosition(202)).* d will be given position value accordingly. Algorithm 3 gives the methodology for insertion of a new context. If d is the first node in a summary, then it is added as a new context, positioned immediately after the current last context. If the current last context has position number 600 then d will take position number 800 (recollect that we assumed *gap* value as 200). If d is not the first node of a summary, then it is added as a context, following the context in which the parent of d is present. *parent*(d) is the node which precedes d in the summary. If the *parent*(d) has position value 302 then d will take position number 400. If the *parent*(d) has position value 800 (*i.e.,* context node) then d will take position number 1000. Note that all the sentences (including duplicates) from the summaries will be

Algorithm 2 Insertion of a node in the neighbourhood of the context node

Input: Partially constructed ILS, context node(c) and the node to be inserted(d)
Output: ILS with the node inserted as a neighbour to c
i = position(c) + 1
while ((A node is present at position i) AND (sim(c, $nodeAtPosition(i)$) $\geq$ sim(c, d))) **do**
  $i++$
end while
{//Increment the position values of all the neighbouring nodes of c that are having position numbers greater than or equal to i}
$j = countNeighbours(c)$
$m = j$
while $m >= i$ **do**
  $positionOfNodeAt(m) = positionOfNodeAt(m) + 1$
  $m--$
end while
Place d at position i

included in ILS.

5. SUMMARY OF SUMMARY GENERATION

Each node in the ILS is assigned a score that is calculated based on both the importance with respect to the query and its importance across the ILS. A part of our node score calculation is inspired by the method proposed in [11]. A significant role is played by the neighbouring nodes i.e., if a neighbour of the node contains a query relevant information then the node is assigned a positive score even in the absence of query relevant information in it. The *neighbourhood* of a node is the set of all the nodes that have a similarity value above a threshold with the node. *Note that neighbourhood in this section is different from previous section.*

$$X_{q_i}(s) = d\frac{sim(s, q_i)}{\sum_{m \in N} sim(m, q_i)} \quad (2)$$

Equation 2 computes relevancy of a node with a query term. d is the bias factor which lies between 0 and 1, N is the set containing all the nodes of ILS and $sim(i, j)$ is computed as given by Equation 1. The value of $X_{q_i}(s)$ will be *zero* in the absence of q_i in s. It will have a positive value if q_i is present in s. The denominator in Equation 2 will be small if q_i is present is fewer nodes. Therefore if a node contains a query term that is present in fewer nodes, then the value of $X_{q_i}(s)$ will be higher. Conversely, if a node contains a query term that is present in majority of the nodes, then the value of $X_{q_i}(s)$ will be lesser.

$$Y_{q_i}(s) = (1 - d) \sum_{v \in adj(s)} \frac{sim(s, v)}{\sum_{u \in adj(v)} sim(u, v)} w_{q_i}(v) \quad (3)$$

Equation 3 computes the score by considering node scores of neighbours of s. $adj(i)$ is the set of all nodes in N that have similarity value above 0.1 with the node i. The bias factor d is the trade-off between these two equations i.e., Equations 2 and 3. The value of d is determined empirically. If d is chosen close to 1 then more importance is given to the similarity of a node with the query and less importance is given to its neighbours. We experimentally found that

Algorithm 3 Adding a context to the partially constructed ILS

Input: Partially constructed ILS and node(d) that is to be inserted as a context
Output: ILS with d included
if d is the first sentence/node of the summary **then**
  ADD d as a new context node to ILS, following the current last context node and set position(d) = position($CurrentLastContextNodeOfILS$) + gap
else
  $i = \lfloor position(parent(d))/gap \rfloor * gap + gap$
  for each node n starting from position i **do**
    position(n) = position(n) + gap
  end for
  Insert d as a context node and set position(d) = i
end if
for each non context node m **do**
  if $sim(m, d) > sim(m, context(m))$ **then**
    $x = position(m)$
    $y = countNeighbours(context(m))$
    Insert m as a neighbour to d by applying Algo 2
    {//Decrement the position values of all the nodes that were following m in previous context}
    $z = y - x$
    $p = 0$
    while $p < z$ **do**
      $positionOfNodeAt(x + 1) = positionOfNodeAt(x + 1) - 1$
      $x++$
      $p++$
    end while
  end if
end for

$d = 0.85$ is the optimal value. Computation of Equation 3 is repeated till the convergence is achieved in Equation 4.

$$w_{q_i}(s) = X_{q_i}(s) + Y_{q_i}(s) \quad (4)$$

$$W_Q(s) = \sum_{1 \leq i \leq t} w_{q_i}(s) \; + \; \frac{1}{|N|} \sum_{m \in N \;\&\; m \neq s} sim(s, m) \quad (5)$$

The node score for each node with respect to each query term $q_i \in Q$ where $Q = \{q_1, q_2, ..., q_t\}$ is computed using Equation 4. Equation 4 is iterated till the scores converge. Here, $w_{q_i}(s)$ is the node score of node s with respect to query term q_i. Score of a node with respect to a query Q is calculated using Equation 5, $W_Q(s)$ is the summation over node scores calculated with respect to each query term using Equation 4 and second part of Equation 5 is to capture the salience of s's information in ILS. For a given query Q, node scores for each node with respect to each query term are calculated. So, a node will have a high score if:

1. It has information relevant to the query.
2. It has neighbouring nodes sharing query relevant information.
3. It has information that is in majority of nodes.

After assigning scores, the node with the highest score is selected as the first sentence to the summary. The node n_i that has the highest score among the remaining ($|N| - 1$)

nodes, calculated with Equation 6, will be included into the summary. s_j is the node that is in the summary. In a similar fashion, other nodes are selected into the summary. Selection process continues till the target summary size is reached. After completing the selection, sentences in the summary are arranged in the increasing order of their position values (computed during ILS construction).

Our claim is: this rearrangement preserves coherence in the summary generated. Recollect, our assumption that individual summary generated on a small set of documents is coherent. In accordance with this assumption, the *base summary* is coherent. ILS is constructed with this *base summary* as its skeleton. This structure remains intact expect during augmenting a new context. This mechanism ensures logical flow to be preserved during the construction of ILS. This is the reason behind achieving a better flow by rearranging the sentences in the order of their position numbers.

$$\underset{i}{Max}\{\lambda W_Q(n_i) - (1-\lambda)\underset{j}{Max}\{sim(n_i, s_j)\}\} \qquad (6)$$

Equation 6 is inspired by Maximal Marginal Relevance (MMR) [3] approach. λ ranges from 0 to 1. If λ is 1 then the most responsive sentence with respect to query is chosen. If λ is 0 then the least redundant sentence is chosen. If λ is close to 1 then more importance is given to responsiveness and less importance to redundancy. λ has to be appropriately chosen so that a node is selected if it is having information highly relevant to the query (first part of the equation) and if its selection does not add redundancy (second part of the equation) to the already selected set. Note that Equation 6 is used to select nodes into the summary and the original node scores that are calculated using Equation 5 are unaltered. The proposed methodology for summary generation captures both the importance of a sentence with respect to query and its significance across the summaries. In this way, both centrality and centroid methods are integrated while assigning score to a sentence. Therefore, the performance of our system is expected to be good. Experimental results show this fact.

6. EXPERIMENTAL SETUP AND RESULTS

S-SUM requires summaries as its input. A summarizer is required to generate summaries on the subsets of documents on a given topic/query. For this purpose, we have chosen a system called $ESUM$ [16]. $ESUM$ is one of the efficient query specific text summarizer. In fact, any such summarizer can be used to provide the input to *S-SUM*, but we found *ESUM* to be as good as any other summarizer. The performance of *ESUM* on *DUC* 2005 [1] data is close to the best system of *DUC* 2005. In any case, using a better individual summarizer will further improve the performance of *S-SUM*. We compared the results of *S-SUM* with the results of *ESUM* run on the *entire* cluster of documents on a specific topic, using the standard *ROUGE* (Recall-Oriented Understudy for Gisting Evaluation) measure used in the community and found that performance of *S-SUM* is better than *ESUM*.

6.1 Experimental Setup

[1]http://duc.nist.gov

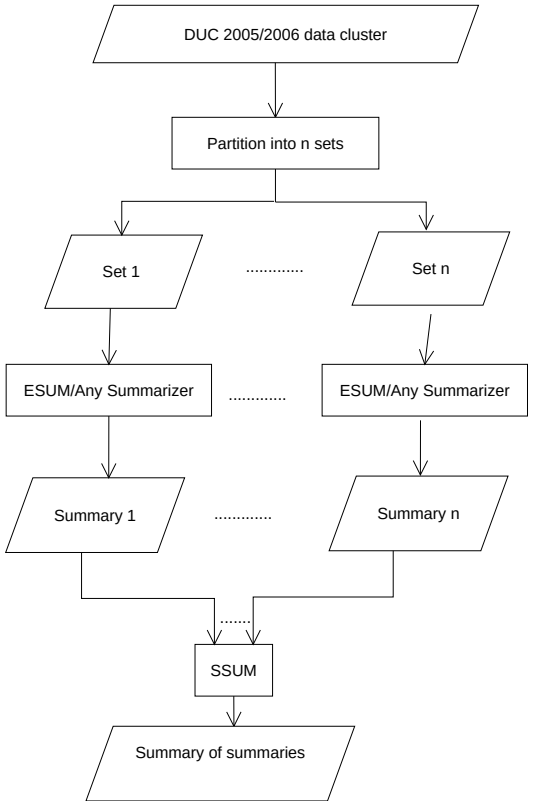

Figure 1: A block diagram of experimental setup

The *DUC* data has 50 clusters and each cluster has number of documents discussing about a particular topic. Figure 1 outlines the detailed experimental setup for a cluster. Documents in a cluster are divided into n disjoint subsets. For example, if we take number of documents in a cluster as 25 and the number of disjoint subsets as 3, then the first 8 documents are assigned to the first set and the next 8 to the second set and the remaining to the third set. Figure 1 illustrates the block diagram of experimental setup. In our experiments, we have chosen the number of sets to be *three*. *ESUM* is used to summarize the documents in each set. After summarization, each set has one summary, called a partial summary[2]. All these partial summaries will be given as an input to *S-SUM*. *S-SUM* summarizes these three summaries. As *DUC* data has 50 clusters, this process is repeated on all the clusters.

To the best of our knowledge, there is no other system in literature which performs the task of generating summary from the summaries. Therefore, we have designed our own methodology to evaluate the performance of *S-SUM*. The summary of summaries generated by *S-SUM* is compared with the summary generated by *ESUM* on the *whole* cluster. We also give *ROUGE* [6] values of summaries of individual sets and the summary generated by *ESUM* on these three summary sets.

6.2 Results

[2]We call it partial because the summary is generated on a subset of the cluster

Table 1: ROUGE Values on DUC 2005 Data

System	ROUGE-1	ROUGE-2	ROUGE-SU4
ESUMC	0.37167	0.07140	0.12768
System-15	0.37515	0.07251	0.13163
S-SUM	0.37733	0.07284	0.13121
SIGIR08	0.35006	0.06043	0.12298
SET1	0.36344	0.06273	0.12054
SET2	0.36444	0.06296	0.12089
SET3	0.35654	0.06301	0.11874
ESUMS	0.35746	0.06384	0.12008

To evaluate the quality of the summaries generated, *DUC* provides us with *ROUGE* [6] values. Also, *DUC* provides model summaries that are written by volunteers. *Recall* value is calculated for the generated summaries with respect to these model summaries. Through *ROUGE* values, we can determine the quality of a summary as these values are computed by comparing the summary with the summaries written by volunteers.

ROUGE-N uses n-gram recall measures of system generated summaries and the summaries generated by the volunteers(model summaries). *ROUGE-N* is calculated as given in Equation 7

$$ROUGE\text{–}N = \frac{\sum_{s \in model\ summaries} \sum_{gram_n \in s} count_{match}(gram_n)}{\sum_{s \in model\ summaries} \sum_{gram_n \in s} count(gram_n)} \quad (7)$$

Here n is the length of a n-gram. $gram_n$ stands for a specific n-gram. $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in both the generated summary and in the reference summaries. *ROUGE-1* is the recall measure of uni-grams. *ROUGE-2* is the recall measure of bi-grams. *ROUGE-SU4* is the recall measure which computes the skip bi-grams with skip distance four and uni-grams are also considered while computing this measure. Finer details of *ROUGE* measurements can be found at [6].

The values of all the variables in the equations are fixed empirically after experimenting on test data of *DUC*. The same set of values are used for both *DUC* 2005 and 2006. The values for bias factor d is 0.85 (based on [11]) and λ is 0.6 (based on [3]). The values in the tables indicate that the generated summaries are consistent and the quality of summaries in terms of the *ROUGE* measures is satisfactory.

All the values in the tables are the *mean* of 50 clusters of DUC. *ESUMC* values are computed on the summaries generated by *ESUM* over the complete set of documents in a cluster. System-15 and System-24 are the best performing systems at DUC 2005 and 2006 respectively. *SIGIR*08 [14] is an efficient summarizer but *ROUGE* values are available for DUC 2005 only. *SETi* values are computed on summaries generated by *ESUM* on the i^{th} set of cluster as discussed earlier. *ESUMS* values are computed on summaries that are generated by summarizing partial summaries generated by *ESUM* (*i.e.,* 3 partial summaries). The *ROUGE* values in the tables clearly demonstrate the performance of *S-SUM* system. Though our system is given only 3 summaries as input, the performance is comparable with the best system of DUC. Note that systems at DUC generate summary on the whole cluster. On contrary $S-SUM$ generates summary on partial summaries. So, the performance

Table 2: ROUGE Values on DUC 2006 Data

System	ROUGE-1	ROUGE-2	ROUGE-SU4
ESUMC	0.38215	0.07514	0.13160
System-24	0.41108	0.09558	0.15529
S-SUM	0.39884	0.08223	0.14037
SET1	0.37846	0.07236	0.12874
SET2	0.37619	0.06910	0.12698
SET3	0.37493	0.07215	0.12835
ESUMS	0.38303	0.07531	0.13248

of DUC systems should be much better than *S-SUM* and in fact it is unfair to compare our system with DUC systems. But the performance of $S-SUM$ is much better than we anticipated. Also, the performance of our system is dependent on the quality of the input summaries. As *ESUM* was close to the performance of the best system, system-15 in DUC 2005, *S-SUM* was able to outperform system-15. This result is in fact a surprise. In DUC 2006, *ESUM* was well behind the best system, system-24, so, *S-SUM* was not able to outperform system-24.

S-SUM generates summary of summaries efficiently and it can also be used as an additional module to the existing summarizers: majority of the summarizers follow an unified approach (all the documents are merged into a single document or inter and intra similarities among the sentences in the documents are taken into consideration) for generating multi-document summaries. To boost the efficiency and performance, the document cluster can be partitioned into multiple sets and a summary can be generated for each set using a summarizer. All these summaries can be given as an input to *S-SUM* to generate summary of summaries and as evident from the results, the quality would be better than the summary generated by that summarizer on the whole cluster. The choice of the partition is by itself a challenging problem. Overall, by partitioning a cluster, both efficiency is achieved and the quality of summary is also improved.

7. CONCLUSIONS

In this paper we have addressed the issue of generating summary of summaries that are generated by extractive summarizers. We have developed a system called *S-SUM* that generates summary from summaries. Integrated Linear Structure(ILS) is introduced by us. *ILS* preserves logical flow between sentences from different summaries. Experimental results demonstrate that the methodology introduced for summary generation produces quality output. Coherence of the summaries is preserved via the position values given to sentences in the *ILS*. Non-redundancy and completeness are achieved by MMR approach. The encouraging experimental results suggest that *S-SUM* can also be used to boost the performance of any extractive summarizer. One limitation of *S-SUM* is it assumes that the base summary is coherent.

8. REFERENCES

[1] Gerard Salton and Amit Singhal and Mandar Mitra and Chris Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, 1997.

[2] L. Page and S. Brin and R. Motwani and T. Winograd. The pagerank citation ranking: Bringing order to the

web. pages, 161–172, Brisbane, Australia, 1998.

[3] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. pages, 335–336, Melbourne, Australia, 1998. ACM.

[4] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[5] Dragomir R. Radev and Hongyan Jing and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. pages, 21–30, Seattle, Washington, 2000. Association for Computational Linguistics.

[6] Chin Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. pages, 605–612, Barcelona, Spain, 2004. Association for Computational Linguistics.

[7] Rada Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. page, 20, Barcelona, Spain, 2004. Association for Computational Lingu.

[8] Güneş Erkan and Dragomir R. Radev. LexPageRank: Prestige in multi-document text summarization. pages, 365–371, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Mihalcea, Rada and Tarau, Paul. In . pages, 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[10] Jagadeesh, J and Pingali, Prasad and Varma, Vasudeva. A relevance-based language modeling approach to duc 2005. In *Proceedings of Document Understanding Conferences (along with HLT-EMNLP 2005), Vancouver, Canada*, 2005.

[11] Jahna Otterbacher and Güneş Erkan and Dragomir R. Radev. Using random walks for question-focused sentence retrieval. pages, 915–922, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics.

[12] Ramakrishna Varadarajan and Vagelis Hristidis. A system for query-specific document summarization. pages, 622–631, Arlington, Virginia, USA, 2006. ACM Press.

[13] Xiaojun Wan and Jianwu Yang and Jianguo Xiao. Manifold-ranking based topic-focused multi-document summarization. pages, 2903–2908, Hyderabad, India, 2007.

[14] Dingding Wang and Tao Li and Shenghuo Zhu and Chris Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. pages, 307–314, Singapore, Singapore, July 2008. ACM.

[15] M Sravanthi and C R Chowdary and P Sreenivasa Kumar. QueSTS: A query specific text summarization system. pages, 219–224, Florida, USA, may 2008. AAAI Press.

[16] C Ravindranath Chowdary and P Sreenivasa Kumar. ESUM: An efficient system for query-specific multi-document summarization. In *ECIR '09: Proceedings of the 31th European Conference on IR Research*, pages, 724–728, Toulouse, France, April 2009. Springer.

[17] C. Ravindranath Chowdary and M. Sravanthi and P. Sreenivasa Kumar. A system for query specific coherent text multi-document summarization. *International Journal on Artificial Intelligence Tools*, 19(5):597–626, 2010.

[18] Alexandra Balahur and Mijail Alexandrov Kabadjov and Josef Steinberger and Ralf Steinberger and Andrés Montoyo. Challenges and solutions in the opinion summarization of user-generated content. *J. Intell. Inf. Syst.*, 39(2):375–398, 2012.

[19] Yin, Wenpeng and Pei, Yulong and Zhang, Fan and Huang, Lian'en. Query-focused multi-document summarization based on query-sensitive feature space. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages, 1652–1656, New York, NY, USA, 2012. ACM.

[20] Zhang, Lanbo and Zhang, Yi and Chen, Yunfei. Summarizing highly structured documents for effective search interaction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages, 145–154, New York, NY, USA, 2012. ACM.