An Incremental Summary Generation System

C Ravindranath Chowdary P Sreenivasa Kumar

Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai
India 600 036
{chowdary, psk}@cse.iitm.ac.in

Abstract

Huge amount of information is present in the World Wide Web and a large amount is being added to it frequently. A query-specific summary of multiple documents is very helpful to the user in this context. Currently, few systems have been proposed for query-specific, extractive multi-document summarization. If a summary is available for a set of documents on a given query and if a new document is added to the corpus, generating an updated summary from the scratch is time consuming and many a times it is not practical/possible. In this paper we propose a solution to this problem. The algorithm finds pair of sentences, one from the current summary and other from the new document, that are to be swapped to improve the quality of the summary. For a given query, quality of a summary is determined by its informativeness, coherence and completeness. We propose a scoring function that captures these features to calculate the quality of a summary. The process of updating/improving summary is continued iteratively till the improvement in quality measure becomes negligible. Our experimental results, both qualitative and quantitative, show that performance of the proposed approach for incremental summary generation is quite encouraging.

1 Introduction

Currently, the World Wide Web is the largest source of information. Huge amount of data is present on the Web and large amount of data is added to the web constantly. Often the information pertaining to a topic is present across several web pages. It is a tedious task for the user to go through all these documents as the number of documents available on a topic will range from tens to thousands. It will be of great help for the user if a query specific multi-document summary is generated.

Summary generation can be broadly divided as abstractive and extractive. In abstractive summary generation, the abstract of the document is generated. The summary so formed need not have exact sentences as present in the document. In extractive summary generation, important sentences are extracted from the document. The generated summary contains all such extracted sentences arranged in a meaningful order. In this paper, generated summaries are extractive. Summary can be generated either on a single document or on several documents. In multi-document summary generation, other issues like time, ordering of extracted sentences, scalability etc. will arise.

Summary can be either generic or query specific. In generic summary generation, the important sentences from the document are extracted and the sentences so extracted are arranged in appropriate order. In query specific summary generation, the sentences are scored based on the query given by the user. The highest scored sentences are extracted and presented to the user as summary.

For a set of documents on a topic and a query related to the topic, suppose a summary is available. If a new document is now made available to the system then summary has to be regenerated with the new document included into the input set by running the query-specific summarizer. But this is not a good solution as it takes considerable amount of time to run the summarizer afresh and a lot of space to store all the original documents. Also, most of the times the documents used for summarization may not be accessible. In our present work, we address the issue of updating the existing summary on the arrival of a new document without having to regenerate the summary from scratch. Specifically, we work under the assumption that the set of initial documents used for generating the current summary are not accessible while updating the summary. To the best of our knowledge this problem is not addressed in the literature.

Contributions of this paper are: 1) We introduce the problem of incremental summary generation 2) A scoring function that measures the quality of a sum-

International Conference on Management of Data
COMAD 2008, Mumbai, India, December 17–19, 2008

mary with respect to a query and the documents summarized is proposed 3) An algorithm is proposed to accomplish the task of incremental summary generation 4) We provide experimental results that prove the effectiveness of the proposed approach.

The rest of the paper is organized as follows: In Section 2 we discuss the related work. Scoring model is discussed in Section 3. In Section 4 Algorithms are discussed. Scoring model is revisited in Section 5. Experimental setup is discussed in Section 6. In Section 7 results are discussed. In Section 8 conclusions and future work are given.

2 Related Work

Text summarization has gained popularity in the recent years. A generic summary generation on single document is discussed by Cajun Wan et al. [20]. Both summary and keywords are extracted from a single document by following iterative reinforcement approach. To extract summary from the document, the following relations are used: sentence-sentence relation, word-word relation and sentence-word relation. A generic summary generation on multiple documents is discussed by Radev et al. in [14]. Centroid based approach is followed by this system, called MEAD, to generate summary. Given a set of documents about a particular topic i.e., a cluster of documents, the centroid of the cluster is calculated. A score is given to each sentence in the cluster with respect to the centroid. Sentences are selected in decreasing order of sentence scores and are arranged with respect to the chronological order of their respective documents.

Abstractive summary generation is discussed in [21, 7, 15]. Extractive summarization is discussed in [14, 3]. A score is given to each sentence based on its importance. In extraction mechanism statistical methods like term frequency, inverse sentence frequency etc. are used. HMM(Hidden Markov Model) is used in CLASSY[3] for giving score to sentences and a pivoted QR algorithm is used to generate a summary. Modeling a document as a graph is proposed in [11]. Here each sentence is modeled as a node[1] and the edges are placed between nodes based on the relatedness. Nodes are given scores by following ideas similar to Pagerank[13] and HITS[6]. Edge scores are calculated based on the amount of similarity between the end points.

Centroid based approach is discussed in [14]. In centroid based summary generation, the centroid is calculated for the document and the sentences in the document are scored with respect to the centroid. Centrality based approaches are discussed in [16, 4, 10, 9]. In centrality based approaches, the salience of a sentence is calculated based on both the contribution of the sentence and the type of neighbouring sentences it is surrounded. Degree centrality is discussed in [16] and eigenvector centrality is discussed in [4, 10, 9]. Concept of bushy path was introduced by Salton et al. in [16]. Nodes with high degree are called bushy nodes. Bushy path is defined as a path connecting top n bushy nodes. 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 [13].

Summary generation for a specific query is discussed in [18, 12, 17]. In [18] topic focused single document summarization is addressed. Document is modeled as a graph, each sentence is considered as a node and an edge is placed between the nodes if the similarity is above a threshold. Irrespective of the threshold, an edge is placed between adjacent sentences in the document. Scores of the nodes are calculated with respect to the given query. A minimal set of nodes that cover all the query terms are picked. If the minimal set doesn't have direct edges among them then intermediate nodes are added to form a connected sub graph of the original graph. In [12] also a document is modeled as a graph. *Cosine similarity* measure is used to calculate the similarity between nodes. Again an edge is present between nodes if the similarity value exceeds a threshold. Node scores are calculated by considering both term frequencies and inverse document frequencies. A node will get high score if it is connected to the nodes with high scores. So, the scores of nodes are computed recursively till the values are converged.

In [19], query is also considered as a node of the document graph. Pairwise similarity between sentences is calculated and an edge is placed between nodes if the similarity value is greater than *zero*. As a query is part of the graph, centrality based approach is followed to select the nodes for the summary. Issue of redundancy is addressed in [5]. Here the graph based model is considered for summarization. Node scores are calculated w.r.t the query. Summary is generated incrementally. A node with highest score is selected into the summary. All the scores of remaining nodes are recalculated based on the nodes already selected into summary and the node score they possess. From the recalculated scores, the highest scored node will be added to summary.

Majority of the approaches mentioned here are highly oriented to summarize news articles. These systems do not explicitly address the efficiency in terms of computation and coherence of the summary generated. They also do not address the problem of updating the generated summary on the availability of a new document. In this paper, we propose a system which gives one of the possible solutions to the problem of incremental summarization.

[1] Here after we use sentence and node interchangeably

3 Scoring Model

3.1 Definitions

1. Complete summary is a summary that contains all the query terms.

2. Coherence is the degree of logical connection between consecutive sentences.

3. Redundancy is the amount of repetitive information present in the summary.

3.2 Node Score

In this section we describe the method to calculate the scores of nodes in a document. A document is modeled as an undirected graph. Each sentence in the document is considered as a node and an edge is present between any two vertices if the *similarity* between the two nodes is above a certain threshold. Before calculating the similarity the stop words are removed and remaining words are stemmed. *Similarity* is calculated as given in Equation 1

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

where $\overrightarrow{n_i}$ and $\overrightarrow{n_j}$ are term vectors for the nodes n_i and n_j respectively.

The weight of each term in $\overrightarrow{n_i}$ is calculated as $tf * isf$. Where tf is $term\ frequency$ and isf is $inverse\ sentential\ frequency$. Number of times a term occurs in a sentence is called as $term\ frequency$. $inverse\ sentential\ frequency$ is defined as $log(\frac{N}{n_t})$, where N is total number of sentences in a document and n_t is number of sentences in which the term is present.

Our node score calculation is inspired by the method proposed by Otterbacher et al. in [12] for calculating the score of a node with respect to a query term. The node scores of the neighbourhood also play a significant role i.e., if the neighbours of a node contain query relevant information then the node is assigned a positive score even if the node does not contain the query term. Here neighbourhood of a node(s) is found out based on the degree of similarity between the nodes. The nodes which have similarity above a threshold(0.1) are all neighbours of s. Initially each node is assigned a query similarity score. The node scores for each node w.r.t each query term $q_i \in Q$ where $Q = \{q_1, q_2, ..., q_t\}$ are computed using the Equation 4. Score of a node w.r.t the query Q is calculated using Equation 5, $W_Q(s)$ is the summation over node scores calculated over individual query terms using Equation 4.

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

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

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

$$W_Q(s) = \sum_{1 \le i \le t} w_{q_i}(s) \tag{5}$$

where $w_q(s)$ is node score of node s with respect to query term q, d is bias factor which lies between 0 and 1, N is the set containing all the nodes of a graph(document) and $sim(i, j)$ is computed as given by Equation 1. Equation 2 computes relevancy of nodes to the query term. Equation 3 computes the score by considering node scores of neighbours of s. The bias factor d gives trade-off between these two equations i.e., Equations 2 and 3 and it is determined empirically. For higher values of d, more importance is given to the similarity of a node to the query when compared to the similarity of a node with its neighbours. We experimentally found out that d value of 0.85 is performing well. For a given query Q, node scores for each node w.r.t each query term are calculated. So, a node will have a high score if

- It has information relevant to the query
- It has neighbouring nodes sharing query relevant information.

In Equation 2 the numerator has bias factor d multiplied with similarity of the node with the query. Denominator has sum of all the node similarity values with the given query. It can be easily seen that the denominator in the equation is for normalization, as evident from Equation 6.

$$\sum_{s \in N} \frac{sim(s, q_i)}{\sum_{m \in N} sim(m, q_i)} = 1 \tag{6}$$

In Equation 3 the similarity value of a neighbouring node(v) of s is calculated and is normalized by dividing it by sum of similarities of all the neighbouring nodes of v. The value so obtained is multiplied by the node score of v calculated with respect to the given query. This calculation is done for all the neighbouring nodes of s and a sum is taken as given in the Equation 3. All the nodes are assigned node scores using Equation 4 and this process is repeatedly done till the node scores converge. Node scores are said to be converged if the difference in node scores in two consecutive iterations is less than a threshold(0.0001).

3.3 Contribution Score

In a given summary, contribution score of a node reflects the actual contribution of that node to the summary. If the node is having information which is already present in other nodes then such node's contribution is less when compared to the node which is having information which is not present in the other nodes. Let $n_1, n_2.....n_n$ be the nodes in the summary. Then N_i is the passage formed by concatenating all the nodes in the summary *except* n_i. Contribution Score $CS(n_i)$ is similarity of n_i with N_i which is calculated using Equation 1. Greater the score, more is the the redundancy. By subtracting the similarity score value from 1, we will get the exact amount of information the node is contributing to the summary. Cumulative Contribution Score(CCS) for summary S is calculated as given in Equation 7.

$$CCS(S) = \sum_{1 \leq i \leq n} (1 - sim(n_i, N_i)) \tag{7}$$

$CCS(S)$ gives the sum of individual contributions of each node with respect to the summary. This measure is very important to find the informativeness and redundancy in the selected set of nodes. $CS(n_i)$ is dependent on the remaining nodes in the summary. $CS(n_i)$ can vary from 0 to 1. Therefore, $CCS(S)$ can vary from 0 to n, where n the number of nodes in the summary.

3.4 Completeness

Completeness is dependent on the query posed by the user. We count the number of query terms present in the given set of nodes. A set which contains all the query terms is 100% complete set. A set which contains partial number query terms is called incomplete set. Here the frequency of the query terms is not considered. Equation 8 gives the $count(S, Q)$, which will be used in Section 3.5. Equation 8 calculates the the number of query terms present in the summary.

$$\begin{aligned} count(S, Q) = {} & No.\ of\ query\ terms\ in\ the\ query \\ - {} & No.\ of\ query\ terms\ not\ present \\ & in\ the\ summary \end{aligned} \tag{8}$$

3.5 Summary Score

The Node Score, Contribution Score and Completeness discussed above determine the quality of the generated summary. In this section we give the methodology for calculating the score of a summary based on these three scores. The equation 9 gives the score for given set of nodes/passage/summary. We use this equation in the Algorithm 1, that is explained in Section 4.

$$\begin{aligned} SScore(S, Q) = {} & \frac{\alpha}{t} * \sum_{s \in S} W_Q(s) \\ & + \frac{\beta}{l} * CCS(S) + \gamma * \frac{count(S, Q)}{t} \end{aligned} \tag{9}$$

The values for α, β and γ are given in Section 7. t is the total number of query terms in the query Q[2]. l is the number of sentences in the summary. Both t and l are for normalization. Summation over $W_Q(s)$ for all the nodes in the summary, will generate aggregate of node scores for the query Q. This measure will give the degree of relatedness between the selected nodes and the query. Higher the value of this summation, more will be relatedness. So, the summation value is directly proportional to the relatedness.

If the value of $CCS(S)$ is more then the contribution of individual nodes in S is more. If the value of $CCS(S)$ is less then the redundancy is more. If both the summation of $W_Q(s)$ and $CCS(S)$ values are high then we can conclude that the set of nodes are significant and information is non-redundant.

The expression $count(S, Q)/t$ gives the fraction of query words present in the set of given nodes. This value ranges from 0 to 1. If this value is 1 then all the query words are present in the nodes and the nodes will form a complete node set. If this value is 0 then the node set does not contain any query term. If the values are high for $W_Q(s)$, $CCS(S)$ and $count(S, Q)/t$ then the set of given nodes are highly query specific, non redundant and complete respectively.

4 Incremental Summary Generation

Recall the problem definition: If a summary is available for a set of documents and if a new document arrives, then we should be able to generate the new summary by considering old summary and the new document alone. The initial set of documents are not considered for generating new/incremental/update summary. Further, the number of sentences in the updated summary should be the same as that of the old one. In this section we give the algorithms to generate the incremental/updated summary.

4.1 Explanation of Algorithms

Algorithm FindExchangePair

The inputs to the algorithm are the current summary, the query and the document from which a node can be selected for inclusion in the current summary. Output is the updated summary and the node in the old summary which got replaced. Current summary and document are considered as sets of nodes. s_1 is the first sentence and s_x is the last one in the summary. Similarly, d_1 is the first sentence in the document and d_y is the last sentence.

In the Lines 10-21, an optimal pair of nodes whose interchange will increase the score of the summary is calculated. If such pair is not present then the output will be the initial current summary itself. The output for this algorithm is the highest scored summary obtained by the replacement of at most one node from

[2] stop words are not considered

Algorithm 1 FindExchangePair

```
Input: Current summary, query and new document
Output: Updated summary and node in summary which got replaced
Let $s_1, s_2, ....s_x$ be the nodes in summary {//No. of nodes in summary = "x"}
Let $d_1, d_2.....d_y$ be the nodes in document {//No. of nodes in document = "y"}
temporarySummary = null
bestSummary = Current summary
maxScore = score of Current summary calculated using Equation 9
temporaryNode = null
x = 1 {// x is the index of document node that is inserted into final summary}
for Each node $d_i$ do
  for Each node $s_j$ do
    temporarySummary = (Current summary - $\{s_j\}) \cup \{d_i\}$
    Calculate Summary Score for temporarySummary using Equation 9
    if Summary Score of temporarySummary > maxScore then
      maxScore = Summary Score of temporarySummary
      bestSummary = temporarySummary
      temporaryNode = $s_j$
      x = i
    end if
  end for
end for
Updated summary = CoherentInsert(bestSummary - $\{d_x\}$, $d_x$)
Return ($Updated\ summary$, $temporaryNode$)
```

the input summary with zero or one node from the new document.

Algorithm IncrementalSummary

Inputs to this algorithm is the currently available summary, query and the new document. Output is the final updated summary. Algorithm FindExchangePair is called till there is no change in the summaries generated in two consecutive calls.

4.2 Maintaining Coherence of Updated Summary

Coherence of a summary is vital in ensuring the readability of the summary generated. In our system we followed the strategy as given in Algorithm CoherentInsert to ensure coherence in the updated summary. This algorithm is called in Algorithm FindExchangePair. Line 6 of the Algorithm CoherentInsert verifies if the maximum similarity node($MaxSimilarNode_k$) is either first or last node of the summary. In either cases the node to be inserted is placed immediately after $MaxSimilarNode_k$. If that is not the case then the as given in Lines 9 and 10, the similarity of the node to be inserted is calculated with the nodes preceding and following the $MaxSimilarNode_k$ in the summary. The maximum between the above two similarities is found and the node is inserted between that and $MaxSimilarNode_k$.

Algorithm 2 IncrementalSummary

```
Input: Current summary, query and new document
Output: Final summary
Let $d_1, d_2.....d_y$ be the nodes in document {//No. of nodes in document = "y"}
CurrentSummary = Current summary
NextSummary = null
tempNode = null
while CurrentSummary $\neq$ NextSummary do
  if NextSummary $\neq$ null then
    CurrentSummary = NextSummary
  end if
  Call FindExchangePair(CurrentSummary, query, new document)
  The return values of previous step are stored in NextSummary and tempNode respectively
  new document = (new document - NextSummary) $\cup$ { tempNode }
end while
Return CurrentSummary
```

4.3 Optimizing Strategy

The algorithm proposed is computationally expensive. It is interesting to reduce the complexity of the algorithm without compromising on the quality of the summary. In the algorithm we consider all the nodes present in the new document. It is observed that many of the nodes are not that prominent in a document. If such unimportant nodes can be eliminated then the efficiency of proposed method will increase. So, we select important nodes form the new document and that selected nodes will be the input to the Algorithm IncrementalSummary. For a given query, every node in the new document is assigned a score using the equation 5. The top scored nodes are selected as the input nodes and the rest are ignored. It is evident that if the new document does not contain nodes related to the query then nodes from it will not be selected for inclusion.

5 Contribution Score Revisited

In Equation 7, $CS(n_i)$ for a node n_i is calculated by finding the similarity of the node w.r.t the rest of the summary and the value so obtained is subtracted from one. Due to the presence of denominator in the Equation 1 for calculating the similarity, we may get simi-

Algorithm 3 CoherentInsert

```
Input: Summary and the node to be inserted
Output: Updated summary
Let $s_1, s_2, ....s_x$ be the nodes in summary
Let $n_i$ be the node to be added
$MaxSimilarNode_k = MAX_{1 \leq j \leq x}(sim(n_i, s_j))$ {// k is assigned to that j for which the maximum similarity is obtained}
if ((k = 1) or (k = x)) then
    Insert $n_i$ after $s_k$ in Summary
else
    $PreNodeSim = sim(n_i, s_{k-1})$
    $ProNodeSim = sim(n_i, s_{k+1})$
    if $PreNodeSim \geq ProNodeSim$ then
        Insert $n_i$ after $s_{k-1}$ in Summary
    else
        Insert $n_i$ after $s_k$ in Summary
    end if
end if
Return Summary
```

larity values close to zero even if the node is repeated in the remaining summary. The former observation is crucial and to overcome this problem, we introduce modified equation for calculating the $CCS(S)$. The $CCS(S)$ is calculated as given in Equation 10.

$$CCS(S) = \sum_{1 \leq i \leq n} (1 - MAX_{i \neq j}(sim(n_i, n_j))) \quad (10)$$

In the Equation 10 the similarity of n_i with the every other node of summary is calculated and the maximum value obtained is taken. If this value is less than 0.7(threshold), then it is subtracted from one. This subtracted value reflects the *maximum* amount of information the node is contributing to the summary. So, more the $CCS(S)$ value lesser is the redundancy. If the threshold value is greater than 0.7 we ignore such node. Greater the similarity, lesser will be the contribution of a node to the summary and more will be the redundancy.

6 Experimental Setup

6.1 Experimentation on DUC

We took DUC(Document Understanding Conference) 2006[3] data for experimentation purpose. DUC 2006 has 50 clusters. A cluster has 25 documents about a topic. 15 documents were chosen and summarized with MEAD[14] system. We update the generated summary by passing documents from 16 to 25 as an input to our model in sequence. The flow chart of the experimental setup is given in Figure 1.

MEAD[14] introduced centroid based summarization. It deals with both single and multi-document summarization. MEAD computes a score for each sentence from the given cluster of related documents by considering a linear combination of several features. We have used centroid score, position and cosine similarity with query as features with 1,1,10 as their weights respectively. MMR(Maximum Marginal Relevance) re-ranker, which is provided by MEAD is used for redundancy removal with a similarity threshold of 0.6.

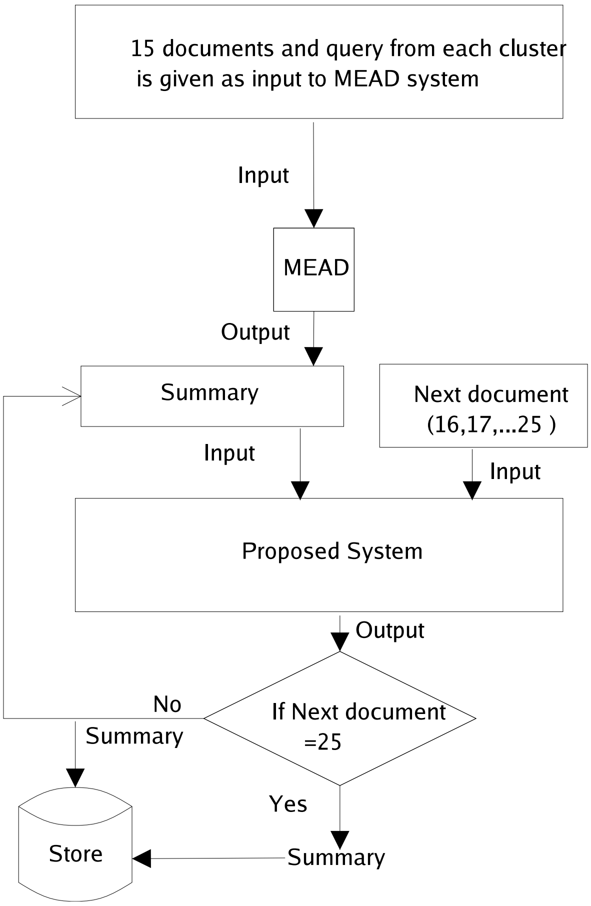
15 documents and query from each cluster is given as input to MEAD system
Input
MEAD
Output
Summary
Next document (16,17,...25)
Input
Input
Proposed System
Output
No
If Next document =25
Summary
Yes
Store
Summary

Figure 1: A block diagram of experimental setup

In the proposed experimental setup as shown in Figure 1, the summary generated by MEAD on first 15 documents in a cluster and the 16^{th} document are the inputs to proposed system. The output of the system is the updated summary. The updated summary generated by the system and the 17^{th} document will be input to our system in the next iteration. The process is repeated till all the documents in a cluster are given as input to our system. Summaries generated after each iteration are stored.

In the proposed model, the sentences are replaced in the current summary if and only if the replacement increases the score of the summary. But while replacing the sentences, the length of updated summary(number of words) will not be same as the length of the summary before updating. So, while evaluating the quality

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

of the updated summary this issue will make the evaluation process very complex. To keep it simple we restrict our summary to 250 words. Also we set summary size option as 250 word in MEAD so that initial summary is of length 250 words. After the updating task, the size of updated summary is calculated and if it less than 250 words then the sentences are added to the summary till the summary size reaches 250 words. The sentences for adding are chosen in the decreasing order of their node scores, calculated using Equation 5.

Another scenario is that the length of updated summary exceeds the 250 word limit. In this case, the sentences are arranged in the decreasing order of their node scores calculated w.r.t the query. The highest scored sentence will be at the beginning of the summary and the lowest scored sentence will be at the end of the summary. When the summaries are given for evaluation only the first 250 words are taken from the summary and the remaining words are discarded.

6.2 Experimentation on CSTBank Corpus

The average query length in the context of search engines is in between 2 and 3 words. The query length in DUC is roughly 20. So we experimented with queries of length 2 and 3 on publicly available database provided by University of Michigan[1]. We experimented on *eight* different topics. Each topic was described in 10 to 14 documents. Each cluster of documents discuss a particular topic like global warming, water on mars etc. The initial summary is calculated on 1^{st} document by MEAD system and remaining documents are added to generate updated summaries. The summary generated by MEAD system on 1 document and the next document will be the input for the generation of updated summary. This process is repeated till the last document. All the summaries generated are stored and they are evaluated by volunteers.

6.3 Discussion on Baseline Summaries

To evaluate the quality of our proposed model, we have to experiment and verify the results. It is very difficult to evaluate such systems. As this problem of update summary generation is proposed for the first time, there is no rival system available to compare the performance. The following alternatives were thought of for a baseline system: 1) Generate a baseline summary using MEAD with all the documents as input. As our system generates the summary by considering only the present summary and new document, this is not a fair comparison. 2) If baseline summary for i^{th} document inclusion is present then baseline summary for $i+1^{th}$ document inclusion can be calculated using MMR approach. But the former approach requires the presence of all the $i+1$ documents to generate a baseline summary. So, it also will not be appropriate baseline.

So we give the ROUGE results generated by the best performing system of DUC 2006(System-24). The ROUGE values are for summaries generated by considering all the 25 documents. But our ROUGE values are *not* obtained by considering all the 25 documents. So, the values of the best system of DUC 2006 would naturally be better than our systems values.

7 Experimental Results

7.1 Results on DUC

The ten updated summaries are evaluated according to DUC 2006 specifications. DUC specifies ROUGE measure to evaluate the quality of summary generated. DUC provides model summaries which are written by volunteers. Recall is calculated for the generated summaries w.r.t this model summaries.

ROUGE[8] stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE has measures to determine the quality of a summary by comparing it to the summaries created by volunteers. ROUGE-N is n-gram recalls between system generated summaries and the summaries generated by the volunteers(model summaries). ROUGE-N is calculated based on the Equation 11

$$ROUGE{-}N = \frac{\sum\limits_{s \in model\ summaries} \sum\limits_{gram_n \in s} count_{match}(gram_n)}{\sum\limits_{s \in model\ summaries} \sum\limits_{gram_n \in s} count(gram_n)} \quad (11)$$

Here n is the length of n-gram. $gram_n$ stands for 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 unigrams. ROUGE-2 is the recall measure of bi-grams. ROUGE-W is the weighted longest common subsequences matching. In longest common subsequence matching, the distance between the words is not given any importance. In weighted longest common subsequence matching weight is given to the distance between the words. Lesser the distance more the weight. 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.

In Table 4 the ROUGE values for updated summaries generated on DUC 2006 are given. The ROUGE values are averaged over 50 clusters of DUC. The values in the table are averaged values over 50 clusters. Updated summary 1 is the summary obtained by updating the summary available for fifteen documents with the sixteenth document. Updated summary 2 is the summary obtained by updating the summary available for sixteen documents with the seventeenth document. The summaries are generated with the following values: $\alpha = 2$, $\beta = 1$ and $\gamma = 3$ (we

Table 1: ROUGE Values on DUC 2006 using Equation 7

Updated Summary	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-SU4
1	0.36993	0.06744	0.08842	0.12246
2	0.36130	0.06260	0.08500	0.11675
3	0.35948	0.05986	0.08440	0.11431
4	0.35694	0.05854	0.08343	0.11334
5	0.35175	0.05627	0.08226	0.11065
6	0.34763	0.05405	0.08079	0.10803
7	0.34420	0.05341	0.08017	0.10698
8	0.34402	0.05195	0.08009	0.10652
9	0.34676	0.05361	0.08118	0.10859
10	0.34545	0.05171	0.08031	0.10694

fixed the values empirically). The values in the Table 4 indicate that the generated summaries are consistent and the quality of summaries in terms of the ROUGE measures is satisfiable.

In Table 3 the ROUGE values for updated summaries generated using the Equation 10 for calculating $CCS(S)^4$ on DUC 2006 are given. The summaries are generated with the following values: $\alpha = 3.5$, $\beta = 1.2$ and $\gamma = 1.5$(we fixed the values empirically). The values in the Table 3 indicate that the updated summaries are consistent and the quality of summaries in terms of the ROUGE measures is satisfiable. Table 2 gives the values of best system in DUC 2006. As the summaries were generated by considering all the 25 documents, these values will be better than ours. But the differences in values are marginal. This shows that performance of our system does not deteriorate much even in the absence of cross document relations.

7.2 Results on CSTBank Corpus

We also experimented on the collection of documents available at University of Michigan[1] and document clusters collected from ProQuest[2]. Each cluster contains 10 to 14 documents. The documents in a cluster discuss particular topics like global warming, water on mars etc. From each cluster a document is selected and a single document summary is generated using a MEAD system with the configuration same as previously mentioned. The summary is updated incrementally by the remaining documents of the cluster. All the intermediate updated summaries are stored.

User evaluation was done on the generated updated summaries. The Table 4 is the averaged evaluation over 2 clusters by five volunteers. V_i represents the i^{th} volunteer. Volunteers were asked to score the summaries on the scale of 5. All the volunteers are graduate students in engineering. They were asked to rank the summary based on the query relatedness and the improvement in the updated summary when compared to earlier summary i.e., updated summary 4 is evaluated both w.r.t the query and updated summary 3. User evaluation results suggest that the proposed method is generating satisfactory summaries.

Table 4: User Evaluation

Updated Summary	V_1	V_2	V_3	V_4	V_5
1	2	2	2	3	2.5
2	2	2.5	2	3	2.5
3	3	3	3	4	3.5
4	4	3.5	3	4	3
5	4	3.5	3.5	4	4
6	4	4	3.5	4	4
7	4	4	3.5	4	4
8	4	4	3.5	4	4
9	4	4	3.5	4	4
10	4	4	3.5	4	4

7.3 Performance Analysis

All the experiments were run on a FC3 with 256MB main memory and 1.7 GHz Intel Pentium processor. The average running time for the generation of an update summary on DUC 2006 corpus is 3.6 minutes/document. In the absence of the complete set of documents, generating a summary close to the quality of the summary generated by the best system of DUC 2006 is a non trivial task. This task is fulfilled by our system efficiently.

8 Conclusions

In this paper we have dealt with updating the available extractive summary in the scenario where the initial documents used for summarization are not accessible. The proposed algorithm updates the available summary as and when a new document is made available to the system. It will replace the sentences from the present summary with the sentences from the new document. Before replacement, our scoring model will check for the best pair of sentences (one from the present summary and another from the newly available document) to be swapped. The pair of sentences that maximizes the score for summary will be chosen for swapping. Our model is designed to consider both the importance of the sentence and its contribution to

[4] only equation is modified there is no change in algorithms

Table 2: ROUGE Values of System-24 on DUC 2006

Updated Summary	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-SU4
System24	0.41108	0.09558	0.11068	0.15529

Table 3: ROUGE Values on DUC 2006 using Equation 10

Updated Summary	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-SU4
1	0.38661	0.08077	0.09380	0.13631
2	0.38581	0.08033	0.09336	0.13566
3	0.38739	0.08159	0.09410	0.13664
4	0.38662	0.08124	0.09382	0.13641
5	0.38685	0.08180	0.09397	0.13667
6	0.38559	0.08079	0.09364	0.13591
7	0.38622	0.08067	0.09371	0.13592
8	0.38559	0.08024	0.09360	0.13560
9	0.38510	0.07997	0.09356	0.13524
10	0.38523	0.07959	0.09347	0.13516

the summary. Experimental results suggest that our model is performing satisfactorily.

9 Acknowledgments

We would like to acknowledge Sravanthi M, Research Scholar for helping in implementation of the proposed system.

References

[1] CSTBank Corpus Available at `http://tangra.si.umich.edu/clair/CSTBank/phase1.htm`.

[2] ProQuest Database Available at `http://proquest.umi.com/login`.

[3] J. M. Conroy, J. D. Schlesinger, and J. G. Stewart. CLASSY query-based multi-document summarization. In *Proceedings of the Document Understanding Conference (DUC-05) at NLT/EMNLP*, Vancouver, Canada, 2005.

[4] G. Erkan and D. R. Radev. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*, pages 365–371, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[5] J. Goldstein and J. Carbonell. Summarization: (1) using mmr for diversity - based reranking and (2) evaluating summaries. In *Proceedings of a workshop on held at Baltimore, Maryland*, pages 181–195, Morristown, NJ, USA, 1998. Association for Computational Linguistics.

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

[7] K. Knight and D. Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press / The MIT Press, July 30 - August 03 2000.

[8] C. Y. Lin and F. J. Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 605–612, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[9] R. Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20, Morristown, NJ, USA, 2004. Association for Computational Lingu.

[10] R. Mihalcea and P. Tarau. TextRank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[11] R. Mihalcea and P. Tarau. Multi-Document Summarization with Iterative Graph-based Algorithms. In *Proceedings of the First International Conference on Intelligent Analysis Methods and Tools (IA 2005)*, McLean, VA, May 2005.

[12] J. Otterbacher, G. Erkan, and D. R. Radev. Using random walks for question-focused sentence retrieval. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 915–922, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.

[14] D. R. Radev, H. Jing, M. Styś, and D. Tam. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938, 2004.

[15] D. R. Radev and K. R. McKeown. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500, 1998.

[16] G. Salton, A. Singhal, M. Mitra, and C. Buckley. Automatic text structuring and summarization. *Inf. Process. Manage.*, 33(2):193–207, 1997.

[17] M. Sravanthi, C. R. Chowdary, and P. S. Kumar. QueSTS: A query specific text summarization system. In *Proceedings of the 21st International FLAIRS Conference*, pages 219–224, Florida, USA, may 2008. AAAI Press.

[18] R. Varadarajan and V. Hristidis. A system for query-specific document summarization. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 622–631, New York, NY, USA, 2006. ACM Press.

[19] X. Wan, J. Yang, and J. Xiao. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, pages 2903–2908, Hyderabad, India, 2007.

[20] X. Wan, J. Yang, and J. Xiao. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559, Prague, Czech Republic, June 2007. ACL.

[21] M. J. Witbrock and V. O. Mittal. Ultra-summarization (poster abstract): a statistical approach to generating highly condensed non-extractive summaries. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 315–316, New York, NY, USA, 1999. ACM.