Document Update Summarization Using Incremental Hierarchical Clustering

Dingding Wang
School of Computer Science
Florida International University
Miami, FL 33199
dwang003@cs.fiu.edu

Tao Li
School of Computer Science
Florida International University
Miami, FL 33199
taoli@cs.fiu.edu

ABSTRACT

Document summarization has become a hot topic in recent years. However, most of existing summarization methods work on a batch of documents and do not consider that documents may arrive in a sequence and the corresponding summaries need to be updated in real time. In this paper, we propose a new summarization method based on an incremental hierarchical clustering framework to update summaries as soon as a new document arrives. Extensive experimental results demonstrate the effectiveness and efficiency of our proposed method.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text clustering*

General Terms

Algorithms, Experimentation, Performance

Keywords

Update Summarization, Incremental Hierarchical Clustering

1. INTRODUCTION

The number of documents on the Internet is continuously increasing with the mass of online sources available, e.g., News and blogs. To help readers to extract their interested information from large numbers of texts efficiently, document summarization has been receiving much attention recently. Most document summarization techniques perform in a batch mode. Given a collection of documents, clustering-based summarization methods usually group the sentences contained in the documents into clusters using certain similarity calculation, and the centroid sentences are selected to form the summary [18, 23, 15, 19]. And graph-based methods are also widely used for document summarization [6, 22]. This type of methods usually constructs a sentence graph, in which each node is a sentence in the document collection, and if the similarity between a pair of sentence is above a threshold or the sentences belong to the same document, there is an edge between the sentence pair. The sentences are selected to form the summaries by voting from their neighbors. However, as popular online publishers generate numerous documents daily, the summaries need to be updated periodically. Thus, updating summaries using current batch document summarization methods requires repeatedly processing previous existing documents, which is time consuming and would cause a waste of operations. In some real applications, e.g., disaster management, the time delay is unacceptable since the disaster evolves quickly and the newest reports need to be summarized in time. To address this issue, we study the problem of updating summaries as soon as new documents arrive.

A potential solution is to design incremental summarization algorithms. One type of straightforward methods is to rank the sentences in the documents according to their scores calculated by a set of predefined features, such as term frequency-inverse sentence frequency (TF-ISF) [18, 14, 24] and number of keywords, which does not involve re-calculating the older sentences. However, these heuristic ranking methods based on simple features could hardly perform well because they assume the sentences are completely independent and existing sentences are not influenced by newly coming sentences at all, which is not practical in many real world scenarios. In this paper, we integrate document summarization techniques into an incremental hierarchical clustering framework to re-organize sentence clusters immediately after new documents/sentences arrive so that the corresponding summaries can be updated efficiently. In the meanwhile, the hierarchical relationship among the sentences can also be displayed and re-constructed in real time.

The rest of the paper is organized as follows. Section 2 discusses the related work of current methods in multi-document summarization and incremental clustering. In Section 3, we introduce our incremental hierarchical clustering based update summarization system. Extensive experimental results are shown in Section 4. Finally Section 5 concludes.

2. RELATED WORK

2.1 Multi-Document Summarization

2.1.1 Traditional Multi-Document Summarization

Traditional multi-document summarization aims to generate a generic or query-focused summary which delivers the

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

major or topic-relevant information of the original document collections. Currently, the most widely used summarization methods are clustering based [18, 14, 23] and graph-ranking based [6, 16, 22]. Clustering-based summarization methods usually perform various clustering techniques on the term-sentence matrices formed from the documents. After the sentences are grouped into different clusters, a centroid score is assigned to each sentence based on the average cosine similarity between the sentence and the rest of the sentences in the same cluster. Finally, the sentences with the highest scores in each cluster are selected to form the summary. Graph-ranking based summarization methods have become more and more popular recently. This type of methods usually constructs a sentence graph, in which each node is a sentence in the document collection, and if the similarity between a pair of sentence is above a threshold or the sentences belong to the same document, there is an edge between the sentence pair. The sentences are selected to form the summaries by voting from their neighbors. Erkan and Radev [6] propose an algorithm called LexPageRank to compute the sentence importance based on the concept of eigenvector centrality (prestige) which has been successfully used in Google PageRank. Other graph-based summarization methods have been proposed in [16] and [22]. Other techniques, e.g., non-negative matrix factorization [17], conditional random file [21], hidden Markov model [3], and latent semantic analysis [9] have also been applied to document summarization.

However, with the rapid growth of documents over the Internet, there is a great necessity to update the existing summaries when new documents arrive. The traditional document summarization methods are not suitable for this task for the following reasons. (1) Most of the methods work in a batch way, thus all the documents need to be process again once new documents come, which causes inefficiency. (2) An alternative is that only newly coming documents are summarized and fused into the existing summary, however, this kind of solution would raise the redundancy issue, which is also hard to deal with.

2.1.2 Update Summarization

TAC recently organizes a special summarization competition called update summarization [1]. The task of TAC update summarization track is that we generate a summary of a multi-document data set based on the assumption that the user has already read a given set of documents. Here the definition of "update summarization" is a little bit different from ours in this paper, since they only consider the newly coming document sets while we work on all the documents at hand. And also in their tasks, they only update the documents once while we assume documents arrive in a sequence, which is more practical in real applications. Since our definition is more general, we can easily adapt our work into the TAC task, and we will show the performance of our adapted method in the experiments.

2.2 Incremental Clustering

There exist many efforts on incremental text clustering algorithms [2, 11, 10], whose main task is to efficiently detect new events or novelty when new documents arrive. In comparison, in this paper, we aim to (1) build a sentence hierarchical tree to fully explore the relationship among events; (2) summarize the whole story of the development of events at current timestamp.

3. INCREMENTAL HIERARCHICAL CLUSTERING BASED DOCUMENT UPDATE SUMMARIZATION

3.1 Framework

Figure 1 demonstrates the framework of our approach. First of all, the sequence of documents is preprocessed. Then an incremental hierarchical clustering method is performed to obtain the sentence hierarchy, and a sentence selection scheme is conducted to obtain the most representative sentence at each tree node. Finally based on the requirement of summary length, the summary at current timestamp is created.

3.2 Preprocessing

Given a collection of documents, we first decompose them into sentences. Then the stop-words are removed and words stemming is performed. After these steps, a sentence-term matrix is constructed and each element is the term frequency.

3.3 Incremental Hierarchical Sentence Clustering (IHSC)

In our update summarization system, we use an incremental hierarchical clustering (IHC) method to build the sentence hierarchy of the document collections. The potential benefits of the IHC method are: (1) As an incremental algorithm, the method can efficiently process the dynamic documents in the sense that new documents are continuously added to the data set. (2) A hierarchy is built to facilitate users to explore the structure and relationship between the sentences. (3) The number of clusters is not pre-defined so that users can cut the hierarchy tree at any level based on their needs.

3.3.1 The COBWEB algorithm

The IHC method used in this paper is COBWEB proposed by Fisher [7, 8], which is one of the most popular incremental hierarchical clustering algorithms. The algorithm performs in a top-down fashion to create a concept tree where each node refers to a concept and contains a probabilistic description of that concept.

The Cobweb algorithm operates based on a heuristic measures called *Category Utility (CU)* as the criterion function to determine the partitions in the hierarchy. Give a partition $\{C_1, C_2, ..., C_K\}$, CU is defined as:

$$\frac{\sum_{k=1}^{K} P(C_k) \sum_i \sum_j [P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2]}{K} \quad (1)$$

where $A_i = V_{ij}$ is an attribute-value pair and C_k is a cluster [7]. CU can be interpreted as the increase in the expected number of corrected guesses of attribute values given the partition$\{C_1, C_2, ..., C_K\}$ over the expected number of correct guesses without such knowledge.

When a new element comes, the COBWEB algorithm traverses the tree top-down starting from the root node. At each node, the COBWEB algorithm performs one of the four possible operations based on the criteria of maximizing

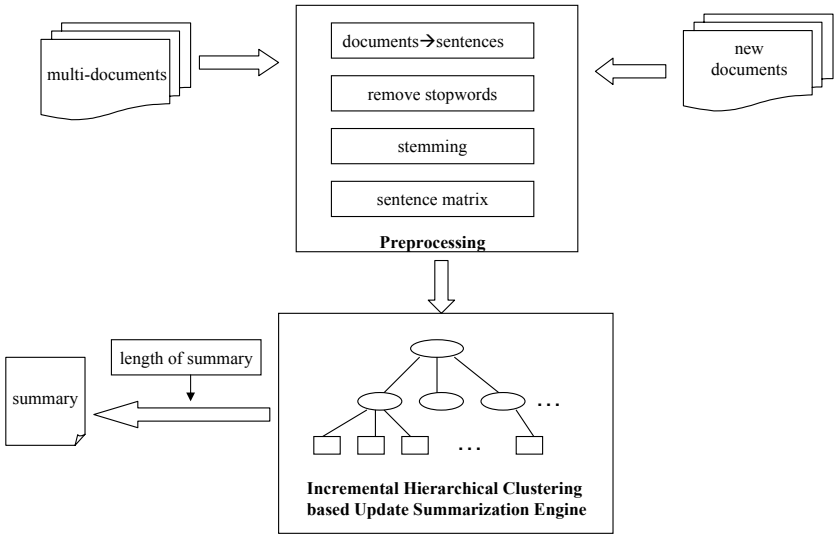

Figure 1: Overview of our approach.

the CU scores. (1)***Insert:*** add the sentence into an existing cluster. (2)***Create:*** create a new cluster. (3)***Merge:*** combine two clusters into a single cluster. (4)***Split:*** divide an existing cluster into several clusters.

3.3.2 COBWEB for text

Sahoo et. al. [20] claim that the original COBWEB algorithm using normal attribute distribution is not suitable for text data, and suggest to use Katz distribution [12] as the word occurrence distribution. Documents are usually represented in the "bag of words" form where terms are attributes, and it has been shown that Katz's distribution works better than other distributions on text data. Thus, [20] proposes a new CU calculation using Katz's distribution as follows.

In Katz's model, assuming word i occurs k times in a document, then

$$p_0 = P(0) = 1 - Pr(the word occurs in a document) \quad (2)$$

and

$$P(k) = (1 - p_0)(1 - p)p^{k-1}, k > 0. \quad (3)$$

From Equation 1, for each attribute i, we need to compute

$$\sum_{f=0}^{\inf} P(A_i = f|C_k)^2 \quad (4)$$

where the attribute value $f = V_ij$ takes 0, 1, ... as the word occurrence count. Substituting Equation 2 and 3 into Equation 4, then

$$\sum_{f=0}^{\inf} P(A_i = f|C_k)^2 = \frac{1 - 2p_0(1 - p_0) - p(1 - 2p_0)}{1 + p} \quad (5)$$

where p_0 and p can be estimated using MLE estimation.

In this paper, we follow the above Katz's distribution based COBWEB algorithm in [20] to create the sentence hierarchical tree incrementally.

3.4 Representative Sentence Selection for Each Node of the Hierarchy

In our update summarization system, we try to select the most representative sentences to summarize each node and its subtrees during the process of the hierarchy generation. Note that the leaves in the hierarchy tree are represented by themselves since each leaf node is one sentence. Once a new sentence arrives, the sentence hierarchy is changed by either of the four operations as described in Section 3.3.1, and in the meanwhile, the representative sentences for the affected nodes are dynamically updated in the following way.

- **Case 1:** If insert a sentence into $cluster_k$, then recalculate the representative sentence R_k of $cluster_k$ using

  $$R_k = Argmax_{s_i}((1-\alpha)Sim(query, s_i)+\frac{\alpha}{K}\sum_{i\neq j} Sim(s_i, s_j))$$

  where K is the number of sentences in the cluster and $Sim()$ is the similarity function between pair of sentences. Here we use cosine similarity. α is a parameter, and is set to 0.6 empirically.

- **Case 2:** If create a new $cluster_k$, the newly coming sentence s_{new} represents the new cluster, i.e., $R_k = s_{new}$.

- **Case 3:** If merge two clusters $cluster_a$ and $cluster_b$ into a new cluster $cluster_c$, the sentence obtaining the higher similarity with the query is selected as the representative sentence at the new merged node.

  $$R_c = Argmax_{R_a,R_b}(Sim(query, R_a), Sim(query, R_b))$$

- **Case 4:** If split $cluster_a$ into a set of clusters as $\{cluster_1, cluster_2, ..., cluster_n\}$, remove node a and substitute it using the roots of its subtrees. The corresponding representative sentences are the representative sentences for the original subtree roots $\{R_1, R_2, ..., R_n\}$.

When all the documents/sentences arrive, the hierarchy and the selected sentences can clearly display the structure of the texts, and also users can cut the hierarchy tree at their desired layers to obtain a summary at that height. For example, a user can determine the cutting level based on the length requirement of the summary.

3.5 the Algorithm

The procedure of the hierarchy generation and summary update is listed as Algorithm 1. The *CheckRelevance*() function used in Algorithm 1 is described in Algorithm 2.

Algorithm 1 Incremental Hierarchical Clustering based Update Summarization (IHCUS)

Input: a query/topic the user is interested in
a sequence of documents/sentences

Output: a sentence hierarchy
the updated summary

1: Read one sentence and check if it is relevant to the given topic, i.e., $CheckRelevance(sentence, topic)$;
2: If relevant, initialize the hierarchy tree with the sentence as the root node, otherwise, remove it and read in the next sentence and repeat Step 1 until the root node is formed; (Note: the *CheckRelevance*() is always performed once we read in a sentence)
3: **repeat**
4: Read in the next sentence, start from the root node; if the node is a leaf, go to Step 5. Otherwise, choose one of the following operations with the highest CU scores.
(1)Insert a node and conduct case 1 for summarization;
(2)Create a node and conduct case 2 for summarization;
(3)Merge a node and conduct case 3 for summarization;
(4)Split a node and conduct case 4 for summarization;
5: If a leaf node is reached, create a new leaf node and merge the old leaf and the new leaf into one node, and case 2 and case 3 are conducted;
6: **until** the stopping condition is satisfied.
7: Cut the hierarchy tree at one layer to obtain a summary with the corresponding length.

Algorithm 2 CheckRelevance()

Input: Q: the query-term vector
S: a sentence-term vector

Output: $result = 1$ indicates the sentence is relevant to the query, otherwise 0

1: Calculate $Sim(S, Q) = \frac{S \cdot Q}{||S||||Q||}$.
2: If $Sim(S, Q) > 0$, result=1; Otherwise 0.

4. EXPERIMENTS

4.1 Data Description and Annotation

4.1.1 Data Sets

- ***Hurricane Wilma Releases (Hurricane)***: The dataset is the collection of press releases by Miami-Dade County Department of Emergency Management and Homeland Security during Hurricane Wilma from Oct. 19, 2005 to Nov. 4, 2005. The collection contains approximately 1,700 documents, which have been categorized into 3 phases based on the status of the hurricane: (1) preparation before hurricane Wilma, (2) damage during the growth of Wilma, and (3) the recovery after the hurricane. The approximate numbers of documents obtained at the end of the three phases are 360, 1310, and 1700, respectively.

- ***TAC 2008 Update Summarization Track (TAC08)***: This is the benchmark dataset from the update summarization track of TAC 2008. The data are from AQUAINT-2 corpus, and consist of 48 topics and 20 newswire articles in each topic. The 20 articles are grouped into two clusters A and B, and the task is to summarize articles in cluster B assuming that users have read documents in cluster A.

4.1.2 Data Annotation

In order to evaluate the quality of the generated summaries by different methods, we use human generated summaries as references. For Hurricane data, we hire 10 human labelers to manually create summaries based on the documents released until the end of each hurricane phase. There are totally 10 queries obtained from the most frequent questions asked in a hurricane disaster management system. Each query is given to 5 different annotators, and each annotator is asked to read all the reports released until the end of each hurricane phase and create three 100 words summaries for each query. For TAC08 data, since it is a benchmark data from Text Analysis Conference (TAC), the human summaries are already provided by the conference. There are four standard summaries written by four human labelers, and these human generated annotations are used for our experimental evaluation.

4.2 Baselines

We implement the following widely used multi-document summarization methods as the baseline systems. Since some of the systems are designed for generic summarization, for fairness we filter the sentences which are not relevant to the given queries by calculating the cosine similarity between the sentences and the queries. And since these systems are designed to work in a batch mode, we run these baseline systems one more time as a new document arrives.

- **Random**: The method selects sentences randomly for each document collection.

- **Centroid**: The method applies MEAD algorithm [18] to extract sentences according to the following three parameters: centroid value, positional value, and first-sentence overlap.

- **LexPageRank**: The method first constructs a sentence connectivity graph based on cosine similarity and then selects important sentences based on the concept of eigenvector centrality [6].

- **LSA**: The method performs latent semantic analysis on terms by sentences matrix to select sentences having the greatest combined weights across all important topics [9].

4.3 Evaluation Measures

In the evaluation, we will compare the results by different methods with the human created summaries using the following evaluation measures.

ROUGE toolkit To compare with the human summaries, we use ROUGE [13] toolkit (version 1.5.5), which is widely applied by Document Understanding Conference(DUC) for document summarization performance evaluation. It measures the quality of a summary by counting the unit overlaps between the candidate summary and a set of reference summaries. Several automatic evaluation methods are implemented in ROUGE, such as ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-SU. ROUGE-N is an n-gram recall computed as follows.

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{ref}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{ref}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \quad (6)$$

where n is the length of the n-gram, and ref stands for the set of the reference summaries. $\text{Count}_{\text{match}}(\text{gram}_n)$ is the maximum number of n-grams co-occurring in a candidate summary and the reference summaries, and $\text{Count}(\text{gram}_n)$ is the number of n-grams in the reference summaries. ROUGE-L uses the longest common subsequence (LCS) statistics, while ROUGE-W is based on weighted LCS and ROUGE-SU is based on skip-bigram plus unigram. Each of these evaluation methods in ROUGE can generate three scores (recall, precision and F-measure). As we have similar conclusions in terms of any of the three scores, for simplicity, in this paper, we only report the average F-measure scores generated by ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU to compare our proposed method with other implemented systems. Intuitively, the higher the ROUGE scores, the similar the two summaries.

4.4 An Illustrative Example

First, we show an example using the Hurricane data. In this example, the query is "What are the situations of Miami-Dade County services including public transit, airport information and public schools". For better illustration in figures, we sampled 150 reports from the original dataset in this example. These documents are released in time sequence and Figures 2 $\sim$ 4 show the hierarchical trees and sample sentences selected at the end of each phase respectively.The three-sentence summaries for the documents released until the end of each time phase are listed in Table 1.

From the sentence hierarchical trees and the generated summaries in this example, we have the following observations. (1) The hierarchy dynamically changes once new documents arrive. (2) From the three sentence structures, we clearly observe the evolution of events along with the growth and reduction of the hurricane. (3) At each time, the sentence best reflecting the new information of each cluster/subcluster is selected as the most representative sentence. (4) The created three-sentence summaries can briefly describe the situations asked in the query in each hurricane evolution phase.

Phase 1	A decision has not yet been made as to whether Miami Dade Public Schools will be open on Monday. All Miami-Dade County services continue normal operations. Miami International Airport is open.
Phase 2	Miami-Dade County Public Schools closed through Tuesday. Miami-Dade transit bus and rail service remain closed. Miami International Airport is will remain closed.
Phase 3	Miami-Dade Public Schools will remain closed on Wednesday. Miami-Dade transit bus and rail service remain closed. Miami International Airport has re-opened with limited service.

Table 1: Three-sentence summaries for each phase.

4.5 Experimental Results on Hurricane Data

4.5.1 Overall Summarization Performance

In this set of experiments, we compare our incremental hierarchical clustering based update summarization (IHCUS) method with the implemented baselines on Hurricane data using Rouge toolkit. For each query and the documents collected in each phase, each system generates one summary. Therefore, there are 10 summaries created by each system for each phase, and 30 in total. The scores reported in this section are the average Rouge F-scores for the 10 summaries of each phase. The experimental result are shown in Tables 2 $\sim$ 4.

Systems	R-1	R-2	R-L	R-SU
Random	0.481	0.123	0.457	0.233
Centroid	0.553	0.139	0.520	0.301
LexPageRank	0.569	0.149	0.512	0.305
LSA	0.544	0.137	0.520	0.313
IHCUS	**0.597**	**0.163**	**0.549**	**0.328**

Table 2: Overall performance comparison on phase 1 of Hurricane data.

Systems	R-1	R-2	R-L	R-SU
Random	0.483	0.119	0.453	0.205
Centroid	0.543	0.130	0.517	0.291
LexPageRank	0.560	0.141	0.517	0.298
LSA	0.531	0.122	0.496	0.301
IHCUS	**0.613**	**0.178**	**0.561**	**0.337**

Table 3: Overall performance comparison on phase 2 of Hurricane data.

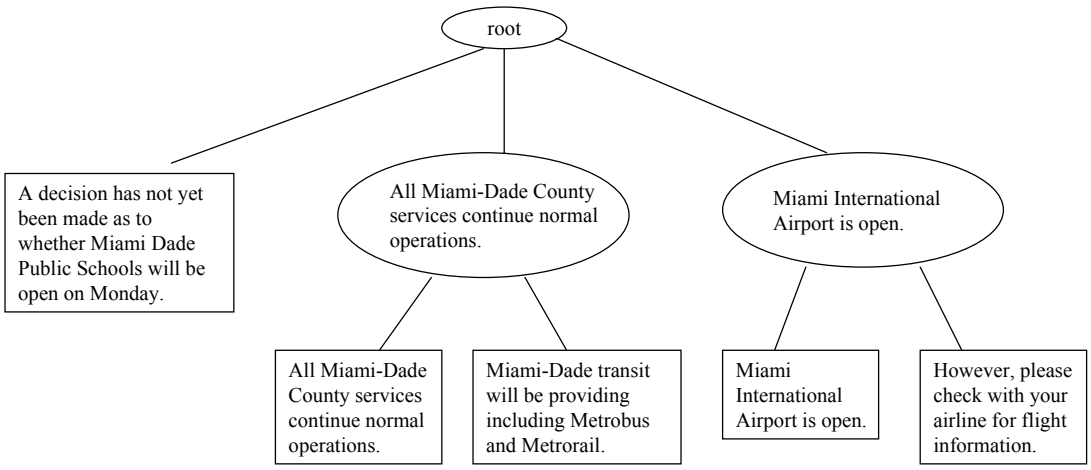

Figure 2: Hierarchy in phase 1.

Systems	R-1	R-2	R-L	R-SU
Random	0.478	0.119	0.445	0.217
Centroid	0.547	0.133	0.511	0.295
LexPageRank	0.558	0.145	0.503	0.301
LSA	0.537	0.125	0.509	0.309
IHCUS	**0.622**	**0.180**	**0.557**	**0.343**

Table 4: Overall performance comparison on phase 3 of Hurricane data using ROUGE evaluation methods.

To better demonstrate the results, Figure 5 visually illustrates the comparison. As we have similar conclusion on different ROUGE scores, we only show the ROUGE-1 results in the figure. And the average scores for all the 30 summaries in all the phases are shown in Figure 6.

From the comparison results, we have the following observations:

- Random has the worst performance.
- The results of LSA are slightly better than those of Random. Note that LSA provides a continuous solution to the most widely used K-means clustering problem while LSA relaxes the non-negativity of the cluster indicator of K-means [4, 5]. Hence this method performs flat clustering-based summarization: it first generates sentence clusters and then selects representative sentences from each sentence cluster.
- The Centroid system outperforms clustering-based summarization methods. This is mainly because the Centroid based algorithm takes into account positional value and first-sentence overlap which are not used in clustering -based summarization.
- LexPageRank slightly outperforms Centroid. This is due to the fact that LexPageRank ranks the sentence using eigenvector centrality which implicitly accounts for information subsumption among all sentences [6].
- All the traditional document summarization methods perform well in the original documents in phase 1, however when new documents arrive, their performance decreases.
- Our IHCUS method outperforms all other implemented baseline systems. The good results benefit mostly from the summaries of phase 2 and phase 3 (Note that we do not list all the individual results for documents in each phase due to the space limit) because the nature of our method can better capture the event change and update. The experimental results provide strong evi-

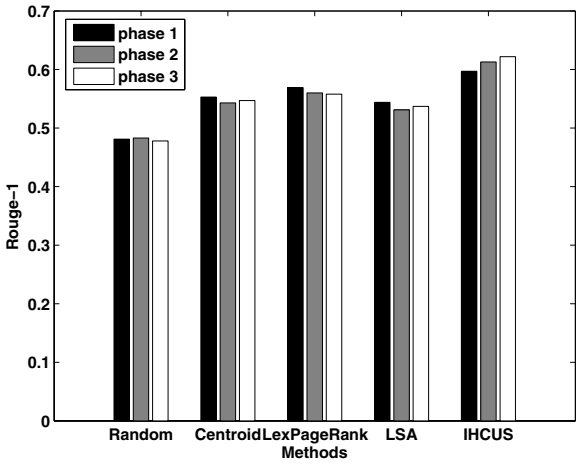

Figure 5: Overall summarization performance on Hurricane Data using ROUGE-1.

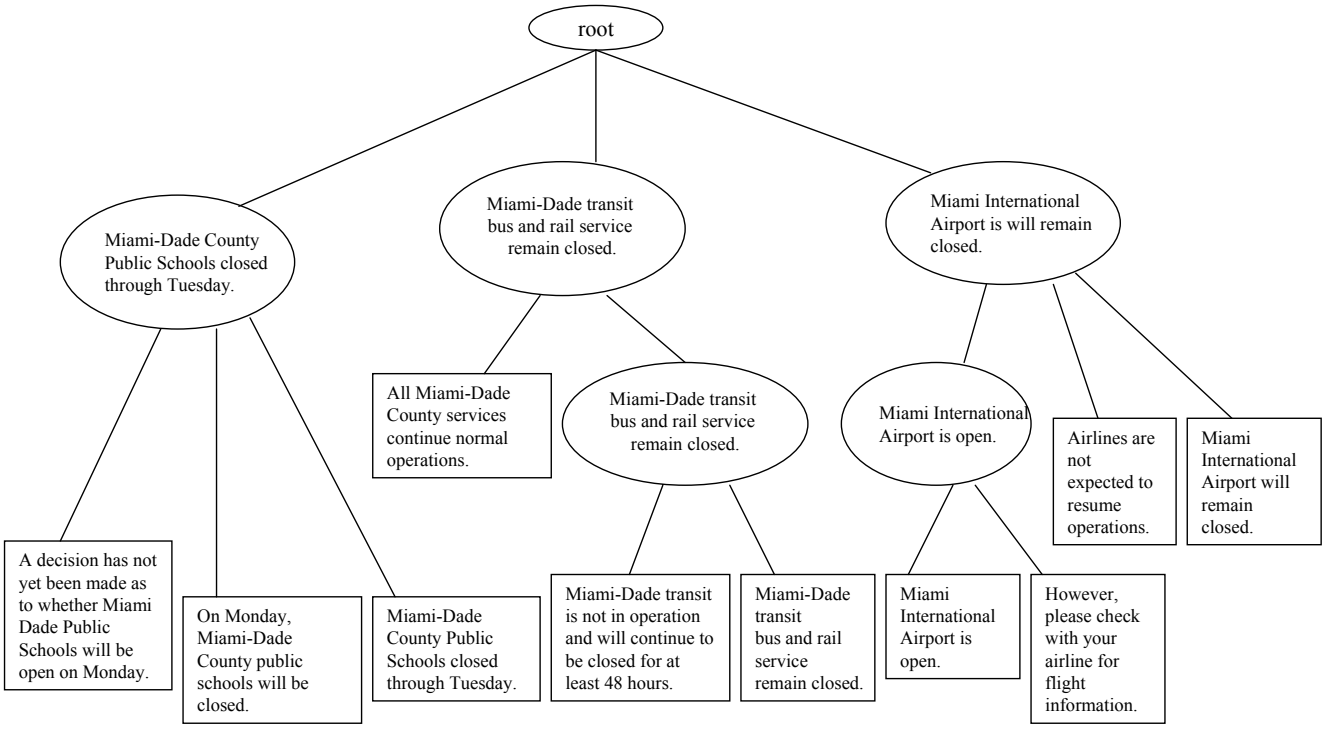

Figure 3: Hierarchy in phase 2.

dence that our IHCUS is a viable method for document summarization when documents arrive in sequence.

4.5.2 Parameter Tuning

Now we examine the weight parameter α used in Case 1 in Section 3.2. When $\alpha = 1$, the sentence most similar to the query/topic is selected. And when $\alpha = 0$, the centroid sentence is selected as the representative sentence. We gradually adjust the value of α from 0.2 to 0.8, and Figure 7 demonstrates the influence of α. Due to the similar observations and space limit, we only show the summarization results on the documents collected at the end of phase 3.

4.5.3 Efficiency

One of the motivations for using incremental hierarchical clustering based summarization method is to efficiently apply dynamic updates to the summaries when new documents come. Therefore, the efficiency of the algorithm computation is an important factor. Table 5 shows the comparison in terms of time spent by each summarizer given one query. Our evaluations are performed by Java running on a Linux machine with quad-core Intel Xeon CPU 2.66GHz and 8Gb memory.

From the experimental results, we clearly observe that (1) traditional centroid-based ranking method performs very slow on large text data; (2) although graph-based summarizer can efficiently generate a summary given a batch of documents, for the update, it still needs to perform the summarization many times, and the total time is accumulated. Since current online data on the web are updated very often, the batch mode algorithms are not able to deal with the frequent updates; (3) the results demonstrate the high efficiency of our system on update summarization.

Systems	phase 1 (min)	phase 2 (min)	phase 3 (min)
Number of summaries	1	2	3
Centroid	23.1	104.7	232.5
LexPageRank	1.7	7.2	17.8
LSA	6.7	23.6	45.3
IHCUS	**1.2**	**3.2**	**4.5**

Table 5: Comparison on time spent for generating summaries in each phase given one query.

4.6 Experimental Results on TAC08 Data

The task of TAC update summarization is to summarize the last 10 documents under the assumption that the reader has already read the first 10 documents. Each summary is no longer than 100 words. As discussed in Section 2, this task is different with the update summarization problem we defined in this paper since in our definition, we do not require the

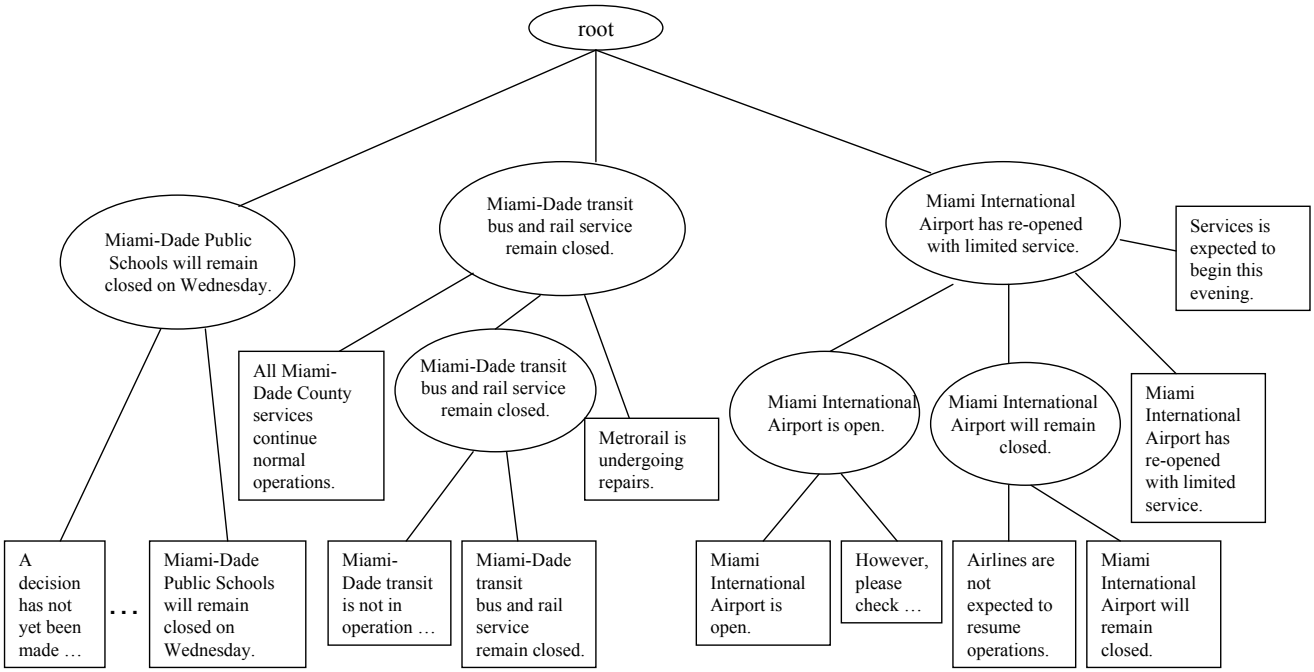

Figure 4: Hierarchy in phase 3.

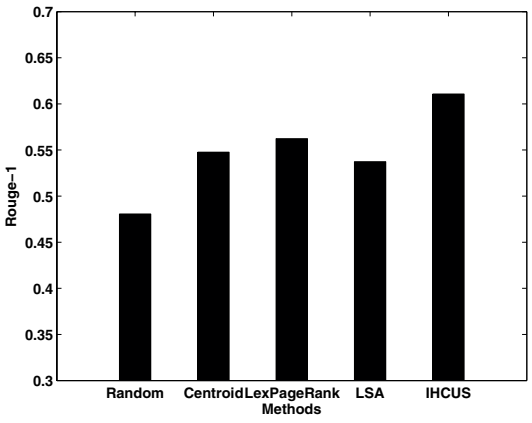

Figure 6: Average summarization performance on Hurricane Data using ROUGE-1.

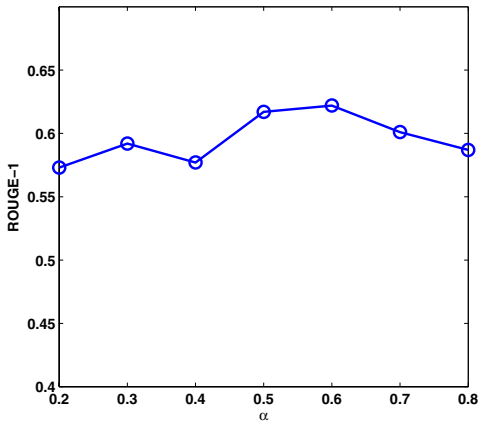

Figure 7: Weight Parameter Tuning on Hurricane Data using ROUGE-1.

pre-reading assumption. However, our method can easily adapt to the TAC task by only selecting the sentences from the last 10 documents. Thus in this set of experiments, we compare our method for the TAC task with the implemented baselines and also the TAC participants. To produce the update summaries in TAC tasks, most of the participants use anti-redundancy techniques because of the assumption given in the task. Many teams rank sentences according to the probability of sentence given the query or the pre-defined distance/similarity between sentences and the query. In the meanwhile, centroid based clustering methods are also widely used in this task. Table 6 demonstrates the overall summarization evaluation. Note that TAC creates a baseline summarization method (denoted as TAC Baseline) to select the first few sentences of the most recent document in the relevant document set, and "TAC Best" represents the best results from TAC participants.

Systems	R-1	R-2	R-L	R-SU
TAC Baseline	0.299	0.065	0.265	0.100
TAC Best	0.370	0.091	0.322	0.131
Centroid	0.319	0.070	0.279	0.113
LexPageRank	0.333	0.074	0.293	0.132
LSA	0.301	0.069	0.271	0.107
IHCUS	**0.371**	**0.093**	**0.325**	**0.135**

Table 6: Overall performance comparison on TAC08 data with TAC update summarization task.

From the results, we confirm most of the observations in Section 4.5.1, and we also some new findings: our method slightly outperforms the best team in TAC. Since we relax the assumption that users have already read the documents in the first batch of documents, we do not perform any particular post processing on redundancy removing, although our method itself can deal with information updates automatically. And since the best team of TAC performs exhaustive enumeration of sentence combinations in their work, it is not suitable for large-scale online data.

5. CONCLUSION

In this paper, we propose an incremental hierarchical clustering based approach to update document summaries in real time when new documents arrive. Our system generates a sentence hierarchical tree to demonstrate the complete structure of the documents, and in the meanwhile a summary of contents at current time point is created. Comprehensive experiments on real-world disaster management data and TAC benchmark data show the effectiveness and efficiency of our update summarization approach.

6. ACKNOWLEDGEMENTS

The work is partially supported by the FIU Dissertation Year Fellowship and NSF grants IIS-0546280 and HRD-0833093.

7. REFERENCES

[1] http://www.nist.gov/tac/.

[2] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of STOC 1997*.

[3] J. Conroy and D. O'Leary. Text summarization via hidden markov models. In *Proceedings of SIGIR 2001*.

[4] C. Ding and X. He. K-means clustering and principal component analysis. In *Prodeedings of ICML 2004*.

[5] C. Ding, X. He, and H. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of Siam Data Mining*, 2005.

[6] G. Erkan and D. Radev. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP 2004*, 2004.

[7] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, pages 2:139–172, 1987.

[8] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Journal of Artificial Intelligence*, pages 40:11–61, 1989.

[9] Y. Gong and X. Liu. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of SIGIR 2001*.

[10] S. Guha, N. Mishra, R. Motwani, and L. OąŕCallaghan. Clustering data streams. *IEEE Symposium on Foundations of Computer Science*, 2000.

[11] C. Gupta and R. Grossman. A single pass generalized incremental algorithm for clustering. In *Proceedings of SIAM Data Mining 2004*.

[12] S. M. Katz. Distribution of content words and phrases in text and language modelling. *Nat. Lang. Eng.*, pages 2(1):15–59, 1996.

[13] C.-Y. Lin and E.Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of NLT-NAACL 2003*.

[14] C.-Y. Lin and E. Hovy. From single to multi-document summarization: A prototype system and its evaluation. In *Proceedings of ACL 2002*, 2002.

[15] I. Mani. *Automatic summarization*. John Benjamins Publishing Company, 2001.

[16] R. Mihalcea and P. Tarau. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP 2005*.

[17] S. Park, J.-H. Lee, D.-H. Kim, and C.-M. Ahn. Multi-document summarization based on cluster using non-negtive matrix factorization. In *Proceedings of SOFSEM 2007*.

[18] D. Radev, H. Jing, M. Stys, and D. Tam. Centroid-based summarization of multiple documents. *Information Processing and Management*, pages 919–938, 2004.

[19] B. Ricardo and R. Berthier. *Modern information retrieval*. ACM Press, 1999.

[20] N. Sahoo, J. Callan, R. Krishnan, G. Duncan, and R. Padman. Incremental hierarchical clustering of text documents. In *Proceedings of CIKM 2006*.

[21] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. Document summarization using conditional random fields. In *Proceedings of IJCAI 2007*, 2007.

[22] X. Wan and J. Yang. Multi-document summarization using cluster-based link analysis. In *Proceedings of the Thirty-First Annual International SIGIR Conference*, 2008.

[23] D. Wang, T. Li, S. Zhu, and C. Ding. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of SIGIR 2008*.

[24] W.-T. Yih, J. Goodman, L. Vanderwende, and H. Suzuki. Multi-document summarization by maximizing informative content-words. In *Proceedings of IJCAI 2007*, 2007.