NUS at DUC 2007: Using Evolutionary Models of Text

Ziheng Lin, Tat-Seng Chua, Min-Yen Kan,
Wee Sun Lee, Long Qiu and **Shiren Ye**
School of Computing
National University of Singapore
Singapore, 117543
{linzihen|chuats|kanmy|leews|qiul|yesr}@comp.nus.edu.sg

Abstract

This paper presents our new, query-based multi-document summarization system used in DUC 2007. Current graph-based approaches to text summarization, such as TextRank and LexRank, assume a static graph model which does not model how input text emerges. A suitable evolutionary graph model that is related to human writing/reading process may impart a better understanding of the text and improve the subsequent summarization process. We propose a timestamped graph (TSG) model motivated by human writing and reading processes, and show how input text emerges under the construction phase of TSG. We applied TSG on both the main task and update summary task in Document Understanding Conferences (DUC) 2007 and achieved satisfactory results. We also suggested a modified MMR re-ranker for the update task.

1 Introduction

Graph-based ranking algorithms such as Kleinberg's HITS (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998) have been successfully applied in citation network analysis and ranking of webpages. These algorithms essentially compute the weights of graph nodes based on global topological information. The graphs that these algorithms applied on - citation networks and the Web - feature a dynamic and evolutionary model, in which they evolve over timesteps.

Recently, a number of graph-based approaches have been suggested for NLP applications based on ranking algorithms like HITS and PageRank. Erkan and Radev (2004) introduced LexRank for multi-document text summarization. Mihalcea and Tarau (2004) introduced TextRank for keyword and sentence extractions. Both LexRank and TextRank assume a fully connected, undirected graph, where text units are modeled as nodes and similarities as edges. After graph construction, both algorithms use a random walk on the graph to redistribute the node weights. However, LexRank and TextRank assume static text graphs which do not model how the input texts evolve. we believe that by integrating evolving process in the graph model, we can better approximate how documents have been written, and then use the role that a particular sentence played in this writing process to infer its function/importance.

2 System Overview

We implement our system using the MEAD summarization toolkit (Radev et al., 2001) and Clair library from the CLAIR research group. Figure 1 shows the overview of our summarization system. The input to the system is a user query and one or two clusters of documents, depending on whether it is the main task or up-

date task.

1. **Sentence splitter**. Given a cluster of documents, the sentence splitter detects and marks sentence boundaries in each document. In addition, it annotates each sentence with the document ID and the sentence number in the document. E.g., ID XIE19980304.0061 means that this document is from 4 March, 1998, taken from Xinhua News; XIE19980304.0061-14 means the 14^{th} sentence in this document. This is the timestamp used in graph construction phase.

2. **Graph constructor**. The timestamped graph is constructed in this phase, with sentences as nodes and similarities as edges. The graph construction process will be discussed in Section 3.

3. **Sentence ranker**. Given a constructed graph, a sentence ranker applies random walk on the graph to redistribute the weights of the nodes. A topic-sensitive random walk is applied as the task is query-based. After the ranking algorithm converges, the sentences are ranked according to their weights. Random walk will be discussed in Section 4.

4. **Extractor**. Finally, an extractor selects the top-ranked sentences and concatenates them to form the summary. The extractor employs a sentence re-ranker to filter out redundant information. Two different modified versions of Maximal Marginal Relevance (MMR) sentence re-ranker are used, depending on whether it is main task or update task. The modified versions of MMR re-ranker are introduced in Section 5.

3 Timestamped Graph Construction

We believe that a proper evolutionary graph model of text should capture the writing and reading processes of humans. Although such human processes vary widely, when we limit

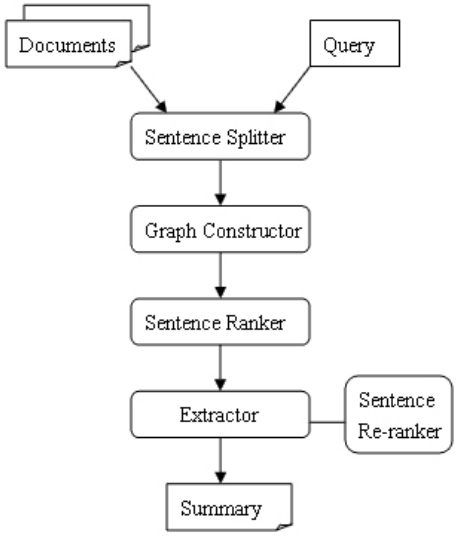


Figure 1: System Overview

ourselves to expository text, we will find that both skilled writers and readers often follow conventional rhetorical styles (Endres-Niggemeyer, 1998; Liddy, 1991). We explore how a simple model of evolution affects graph construction and subsequent summarization. Our work is only exploratory and not meant to realistically model human processes and we believe that deep understanding and inference of rhetorical styles (Mann and Thompson, 1988) will improve the fidelity of our model. Nevertheless, a simple model is a good starting point.

We make two simple assumptions of human writing and reading processes:

1. Writers write articles from the first sentence to the last;

2. Readers read articles from the first sentence to the last.

The assumptions suggest that we add sentences into the graph in chronological order: we add the first sentence, followed by the second sentence, and so forth, until the last sentence is added.

These assumptions are suitable in modeling the growth of individual documents. However when dealing with multi-document input, our

assumptions do not lead to a straightforward model as to which sentences should appear in the graph before others. One simple way is to treat multi-document problems simply as multiple instances of the single document problem, which evolve in parallel. Thus, in multi-document graphs, we add a sentence from each document in the input set into the graph at each timestep. Our model introduces a skew variable to model this and other possible variations.

The pseudocode in Figure 2 summarizes how we would build a timestamped graph for multi-document input set. Informally, we build the graph incrementally, introducing new sentence(s) as node(s) in the graph at each timestep. Next, out of those not yet connected with it, each sentence in the graph picks and connects to the one that is most similar to itself. This process continues until all sentences are placed into the graph.

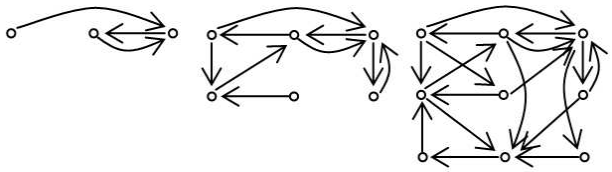
Input: **M**, a cluster of m documents relating to a common event;
Variables:
i = index to sentences, initially 1;
G = the timestamped graph, initially empty.
Loop:

1. Add the i^{th} sentence of all documents into **G**. If a document is shorter than i, no sentence is added for this document.
2. Let each existing sentence in **G** choose and connect to **one** other existing sentence in **G**. The chosen sentence must be one which has not been previously chosen by this sentence.
3. **if** there are no new sentences to add, break; **else i**++, go to Step 1.

Output: **G**, a timestamped graph.

Figure 2: Pseudocode for a specific instance of a generic timestamped graph construction algorithm

Figure 3 shows an example of the graph building process over three timesteps, starting from an empty graph. Assume that we have three documents and each document has three sentences. Let $d_x s_y$ indicate the y^{th} sentence in the x^{th} document. At timestep 1, sentences $d_1 s_1$, $d_2 s_1$ and $d_3 s_1$ are added to the graph. Three edges are introduced to the graph, in which the edges are instantiated by some strategy; say, by choosing the destination sentence by its maximum cosine similarity with the sentence under consideration. Without loss of generality let us say that this process connects $d_1 s_1 \rightarrow d_3 s_1$, $d_2 s_1 \rightarrow d_3 s_1$ and $d_3 s_1 \rightarrow d_2 s_1$. At timestep 2, sentences $d_1 s_2$, $d_2 s_2$ and $d_3 s_2$ are added to the graph and six new edges are introduced to the graph. At timestep 3, sentences $d_1 s_3$, $d_2 s_3$ and $d_3 s_3$ are added to the graph, and nine new edges are introduced.



(a) Timestep 1 (b) Timestep 2 (c) Timestep 3

Figure 3: An example of the growth of a timestamped graph.

The above illustration is just one instance of a timestamped graph with specific parameter settings. We generalize and formalize the timestamped graph construction algorithm as follows:

Definition: A timestamped graph construction algorithm $tsg(M)$ is a 9-tuple $(d, e, u, f, \sigma, t, i, s, \tau)$ that specifies an algorithm that takes as input the set of texts M and outputs a graph G, where:

- d specifies the direction of the edges, $d \in \{f, b, u\}$, which represents forward/backward/undirected edges. It specifies the direction of information flow;
- e is the number of edges to add for each vertex in G at each timestep, $e \in \mathbb{Z}^+$;
- u is 0 or 1, where 0 and 1 specifies unweighted and weighted edges, respectively;
- f is the inter-document factor, $0 \leq f \leq 1$. If intra-document edges are preferred over inter-document edges, we replace the weight w for inter-document edges by fw;

σ is a vertex selection function $\sigma(u, G)$ that takes in a vertex u and G, and chooses a vertex $v \in G$. One strategy is to choose v that has the highest similarity with u and has not yet been chosen by u in previous iterations. There are many similarity functions to use, including token-based Jaccard similarity, cosine similarity, or more complex models such as concept links (Ye et al., 2005);

t gives the granularity of the text units, $t \in \{word, phrase, sentence, paragraph, document\}$. In the task of automatic text summarization, systems are conveniently assessed by letting text units be sentences;

i is the node increment factor, $i \in \mathbb{Z}^+$. It means that at each timestep, i vertices are added for each document;

s is the skew degree, $s \geq -1$ and $s \in \mathbb{Z}$, where -1 represent free skew and 0 no skew. If document d_1 is earlier than d_2 in time, $s = 1$ means that the first sentence of d_1 is added 1 timestep earlier than that of d_2;

τ is a document segmentation function $\tau(\bullet)$. If the documents vary widely in their lengths, we may need to segment long documents so that they have similar lengths.

Following our definition, the example in Figure 3 can be succinctly represented by the tuple $(f, 1, 0, 1, max{-}cosine{-}based, sentence, 1, 0, null)$. For a more detailed description of timestamped graph construction and skew degree we refer the readers to (Lin and Kan, 2007).

4 Sentence Ranking

Once a timestamped graph is built, we want to compute an importance score for each node. The graph G shows how information flows from node to node, but we have yet to let the information actually flow. Most graph algorithms use an iterative method that allows the weights of the nodes to redistribute until stability is reached. One method for this is by applying a random walk, used in PageRank (Brin and Page, 1998). In PageRank, the Web is treated as a graph of webpages connected by links. It assumes users start from a random webpage, moving from page to page by following the links or jumping randomly.

The original equation of PageRank assumes unweighted edges in computation as it assumes hyperlinks are unweighted. In our text graphs, unweighted edges may cause loss of information. As we have a query for each document cluster, we also wish to take queries into consideration in ranking the nodes. Haveliwala (2003) introduced a topic-sensitive PageRank computation. The key to creating topic-sensitive PageRank is that we can bias the computation by restricting the user to jump only to a random node which has non-zero similarity with the query. Integrating weighted edges and query-as-topic sensitivity into the original PageRank equation, we compute a weight $PR(u)$ for node u according to:

$$PR(u) = \alpha \frac{sim(u, Q)}{\sum_{y \in N} sim(y, Q)} + (1 - \alpha) \sum_{v \in In(u)} \frac{w_{vu}}{\sum_{x \in Out(v)} w_{vx}} PR(v) \quad (1)$$

Here N is the set of all nodes in the graph, $sim(u, Q)$ is the similarity score between node u and the query Q. $In(u)$ is the set of nodes that point to u, $Out(v)$ is the set of nodes that node v points to and w_{vu} represents the weight of the edge pointing from v to u. The α is a balancing factor that can be set between 0 and 1. Equation 1 is applied on a timestamped graph G iteratively until it converges. The weights of the nodes are ranking scores of the sentences.

5 Sentence Extraction

The original MMR re-ranker proposed by Carbonell and Goldstein (1998) defines that a document has MMR if it is relevant to the query and contains minimal similarity to previously selected documents. It integrates a negative effect of the maximal similarity of the candidate document and one selected document. Ye *et al.* (2005) revise the equation and introduce a modified version of MMR sentence re-ranker as shown in Equation 2. They show that con-

sidering the maximal similarity of a candidate sentence and a selected sentence is not always optimal, and propose considering the total similarity of the candidate sentence with all selected sentences in the sentence selection stage.

$$MMR_{mod} = \underset{s_i \in R-S}{\text{argmax}}[\lambda \cdot Score(s_i) + (1-\lambda) \cdot sim(s_i, Q) - \delta \cdot \sum_{s_k \in S} sim(s_i, s_k)]. \quad (2)$$

In Equation 2, $Score(s)$ represents the ranking score for sentence s computed in Section 4; λ represents a tuning factor between a sentence's importance and its relevance to the query; δ is the penalty factor which is used to decrease the rank of sentences that are similar to the already selected summary sentences in S. Before sentence extraction, the set of selected sentences, S, is empty; as sentences are extracted, they are also added to S.

While the above is fine for the traditional multi-document summarization task, it does not model the update summary task well. We need to differentiate between sentences in the two different clusters. We propose a modified version of MMR in Equation 3 that is applicable in the update task. Under the assumption for update summary task that readers already read the previous cluster(s), we do not want to select sentences that have redundant information with previous cluster(s). Thus, during the computation of MMR in the update task, not only selected sentences in the summary are under consideration, but sentences in previous cluster(s) also need to be taken into account. To speed up the computation of total similarity of the candidate sentence and the sentences in previous cluster(s), and to remove noisy sentences from previous cluster(s) from the computation, we only consider P, which contains some top-ranked sentences in previous cluster(s). Here γ is the penalty factor that is similar to δ in effect.

$$MMR_{mod2} = \underset{s_i \in R-S}{\text{argmax}}[\lambda \cdot Score(s_i) + (1-\lambda) \cdot sim(s_i, Q) - \delta \cdot \sum_{s_k \in S} sim(s_i, s_k) - \gamma \cdot \sum_{s_j \in P} sim(s_i, s_j)] \quad (3)$$

Assume that readers already read all documents in cluster A, and now we need to extract a summary for cluster B for the readers. Suppose we already have the ranked list of sentences of cluster A (if we don't, we need to compute it using sentence ranking), we add some top-ranked sentences from A into the set P. During sentence extraction, the sentence under consideration will not only be compared to selected sentences in S, but also be compared to sentences in P.

6 Summarization Processes

For the main task, there are 45 document clusters, each input cluster contains 25 documents and a user query, and the system needs to generate a 250-word summary for each cluster. The summarization process for the main task is straightforward. We construct a timestamped graph for the input cluster in the graph construction phase. In sentence ranking phase, we run topic-sensitive PageRank to smooth the weights of the nodes (scores of sentences), followed by ranking the sentences by their scores in decreasing order. In sentence extraction phase, Equation 2 is applied to the ranked list to minimize redundancy during the extraction of summary sentences.

Ten documents clusters are selected from the 45 clusters of the main task for preparation of the update summary task. Each of these ten clusters is divided into three smaller clusters, A, B and C. Usually A contains ten documents, B eight documents and C seven documents. The three smaller clusters have the same user query as the original larger cluster. The summarization system needs to give a 100-word update summary for each smaller cluster. Documents in cluster A are published earlier than that in B, and documents in B earlier than that in C. We apply the same summarization process for cluster A as in the main task to extract a summary for cluster A, with the exception that we use Equation 3 in sentence extraction phase, instead of Equation 2. In Equation 3, P is defined as the set of top-ranked sentences in the clusters that are read by the readers previously.

Notice that during sentence extraction here, P is empty as there is no previously-read cluster before cluster A. After the 100-word summary is generated, twenty top-ranked sentences in A are selected to be included in P, so that P can later be used in the summarization process of cluster B.

To summarize the cluster B, we construct a timestamped graph for all documents in both clusters A and B in graph construction phase. In sentence ranking phase, we then run topic-sensitive PageRank on the graph, followed by ranking the sentences in cluster B by their scores, ignoring sentences in cluster A (i.e., all sentences belong to A receive zero score). In sentence extraction phase, we apply Equation 3 to the ranked list, where P is from the summarization of cluster A. After a 100-word summary is extracted, twenty top-ranked sentences in B are added into P. P now contains 40 top-ranked sentences, in which twenty are from cluster A and the other twenty from clusters B, and is ready for the summarization of cluster C.

To summarize the cluster C, we construct a timestamped graph for all documents in all clusters A, B and C, and then run topic-sensitive PageRank on the graph. We rank the sentences in cluster C by their scores, ignoring sentences in clusters A and B. We apply Equation 3 to the ranked list during sentence selection, where P is from the summarization of cluster B. A 100-word summary is generated for cluster C.

7 Experimental Results

For the main task, our system uses the tuple $(u, 1, 1, 1, concept{-}link{-}based, sentence, 1, 0, null)$ in timestamped graph construction, applies topic-sensitive PageRank in sentence ranking, setting $\lambda = 0.8$ and $\delta = 6$ for Equation 2 in sentence extraction. The values for λ and δ and tuned from DUC 2005 and 2006 datasets. *concept−link−based* is a node selection function based on the concept link computation used in our previous summarization system. Figure 4 shows ROUGE-2 and ROUGE-SU4 scores for our system. The average recall under ROUGE-2 and ROUGE-SU4 are 0.1037 and 0.15857, respectively, and our systems ranks 12^{th} for ROUGE-2 and 10^{th} for ROUGE-SU4 among all 32 peer systems.

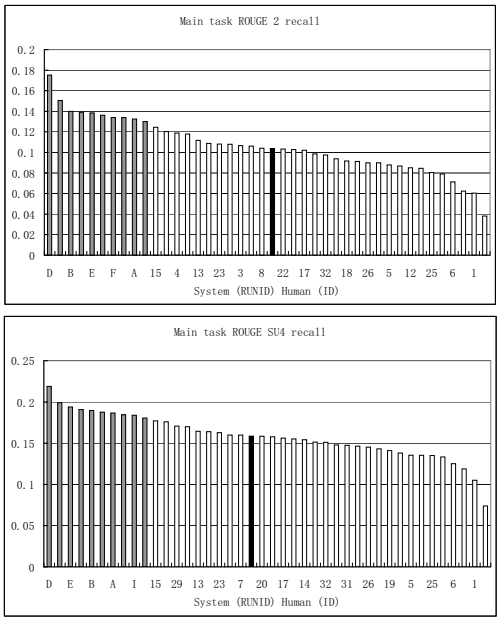


Figure 4: ROUGE scores for main task. Black, gray and white bars denote our system, human annotators and peer systems, respectively.

There are 24 systems participated in the update summary task. Our system uses the tuple $(u, 1, 1, 1, concept{-}link{-}based, sentence, 1, 0, null)$ in timestamped graph construction, applies topic-sensitive PageRank in sentence ranking, and set $\lambda = 0.8$, $\delta = 3$ and $\gamma = 6$ for Equation 3 in sentence extraction. We do not have test data for update summary task to tune these three parameters. We set the values for them based on our experience on the main task. Figure 5 shows ROUGE-2 and ROUGE-SU4 scores for our system. The average recall computed using ROUGE-2 and ROUGE-SU4 are 0.09622 and 0.13245, respectively, and our systems ranks 3^{rd} for ROUGE-2 and 4^{th} for ROUGE-SU4 among all 24 peer systems.

The system performs better in the update task than in the main task. One reason is that the lengths of the summaries are shorter (100 words in update task compared to 250 words in main task), and our system performs well in locating the most topic-related and salient sentences

(the top two or three sentences) but less well in locating the generic and salient ones. Another reason is that the second modified version of MMR as in Equation 3 works well in distilling redundant information that is shown in previously-read cluster(s).

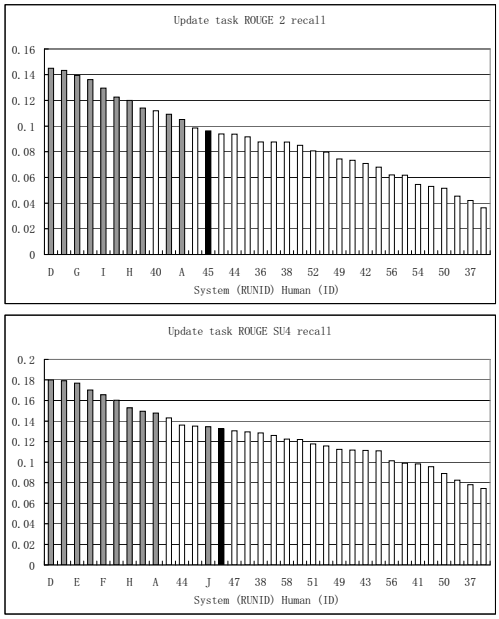


Figure 5: ROUGE scores for update task. Black, gray and white bars denote our system, human annotators and peer systems, respectively.

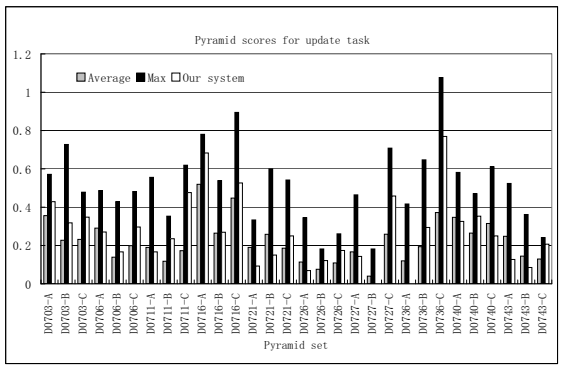


Figure 6: Pyramid scores for update task. Gray, black and white bars denote average scores, maximum scores and scores for our system, respectively.

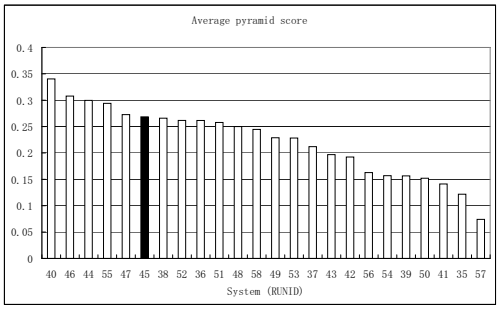


Figure 7: Average pyramid scores for update task. The RUNID for our system is 45.

There are in total 30 standard pyramids created by annotators. Figure 6 shows the average score, maximum score and our system's score for each pyramid set. Our system scores better than the average scores in 18 out of 30 sets. The average scores over all pyramid sets are shown in Figure 7. The best system has the average score of 0.3403, whereas our system obtains 0.2684 on average, which is ranked 6^{th} among all 24 systems.

There are two sets of pyramids that our system obtain zero score. In D0736-A, the most important sentence extracted by our system is a list of talk shows. This sentence has very generic information and thus has a high in-degree in the graph, but is not related to the query. Figure 8 show the update summaries for D0736-A, D0736-B and D0736-C and their query. The texts in bold are information that also shown in some model summaries. We see that there is no bold text in the summary for D0736-A; the sentences that the system extracts are generic but not relevant to the query.

In D0727-B, it seems that information from both D0727-A and D0727-B is not enough to help the system to locate the updated and important information in D0727-B. When we go to D0727-C, the information provided from D0727-A, -B and -C is then enough for the system to outperform the average.

Looking into all 30 sets of pyramid results, we notice that our system always outperforms the average in all clusters C, except D0740-C. The performance in B tends to be better than that in A, and the performance in C better than that in B. This is because in a text graph like timestamped graph, more global information helps the system to locate the updated and salient

Query:
Note the various subjects and controversial incidents on Oprah's show 1998-2000.
D0736-A:
Talk show: "Leeza," NBC; "The Rosie O'Donnell Show," Syndicated; "Live With Regis and Kathie Lee," Syndicated; "The Oprah Winfrey Show," Syndicated; "The View," ABC. The return to violence helped "Springer" claw its way to a virtual tie with one-time giant "The Oprah Winfrey Show" in July ratings as metro Atlanta's most-watched daytime talk show. Vogue's Anna Wintour exults in the editor's note at the front of the Oprah issue that the magazine staff was thrilled "when we heard that Oprah Winfrey wanted to be made glamorous. Oprah Winfrey is fed up with the sleaze on daytime television – especially rival
D0736-B:
The talk show host Oprah Winfrey, a powerful figure in the book world because of Oprah's Book Club, received a 50th Anniversary Medal from the foundation. Talk show host **Oprah Winfrey is starting a magazine next spring with the magazine publisher Hearst Corp.** Hearst said in a statement that the new magazine will offer articles on community, family, relationships, health and fitness, fashion and beauty, and books. The success of "Donahue" makes room for **Oprah Winfrey, who takes her Chicago-based talk show national.** "There's a lot of instant power in being the head of a studio," Tierney noted.
D0736-C:
Under Oprah's influence Tipper would show a touch of girlish soulfulness, while Oprah could add chipper gentility to her repertory. **Oprah is a real-life extension of this fantasy, bestowing gifts on her television audiences - food, CDs, Palm Pilots** - and opportunities for self-enhancement and self-discipline on all her fans. **George W. Bush courts women voters by appearing on the Oprah Winfrey show as Al Gore did last week.** Both in her magazine and on her television show, Oprah's name is used in a queenly manner. What's hers is ours, it seems: **the magazine's sensible psychologist, Dr. Phillip C. McGraw,**

Figure 8: Update summaries for D0736-A, D0736-B and D0736-C. The texts in bold show updated information that corresponds to human summaries.

parts more effectively.

We did not experiment and integrate the skew degree s and document segmentation function τ into the system because of time constraint. Free skew may capture the time order between documents, and applying segmentation on extremely long documents may prevent graphs with dense edges. We believe that once integrated with s and τ, the performance on both tasks will be increased.

8 Conclusion

We have proposed a timestamped graph model for text graph construction. This model is based on our assumptions about human writing and reading processes. As part of DUC 2007, we participate in both the main task and the update summary task. Based on the modified version of MMR re-ranker we used in DUC 2005, we suggested another version of MMR re-ranker for sentence extraction in the update summary task. The results are satisfactory and show that the modified MMR re-ranker works very well in update task. We believe that when integrated with free skewing and document segmentation, our system will give a better performance.

References

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, April.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336.

Brigitte Endres-Niggemeyer. 1998. *Summarizing information*. Springer New York, New York.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Taher H. Haveliwala. 2003. Topic-sensitive PageRank: A context-sensitive ranking algorithm for Web search. *IEEETKDE: IEEE Transactions on Knowledge and Data Engineering*, 15.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, November.

Elizabeth D. Liddy. 1991. The discourse-level structure of empirical abstracts: an exploratory study. *Information Processing and Management: an International Journal*, 27(1):55–81, January.

Ziheng Lin and Min-Yen Kan. 2007. Timestamped Graphs: Evolutionary models of text for multi-document summarization. In *Proceedings of HLT-NAACL 2007 Workshop on TextGraphs-2*, Rochester, April.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July.

Dragomir R. Radev, Sasha Blair-Goldensohn, and Zhu Zhang. 2001. Experiments in single and multi-document summarization using mead. In *Proceedings of ACM SIGIR'01 Workshop on Text Summarization*, New Orleans, Louisiana, September 13.

Shiren Ye, Long Qiu, Tat-Seng Chua, and Min-Yen Kan. 2005. NUS at DUC 2005: Understanding documents via concepts links. In *Proceedings of DUC 2005*.