A Document Graph Based Query Focused Multi-Document Summarizer

Sibabrata Paladhi
Department of Computer Sc. & Engg.
Jadavpur University, India
sibabrata_paladhi@yahoo.com

Sivaji Bandyopadhyay
Department of Computer Sc. & Engg.
Jadavpur University, India
sivaji_cse_ju@yahoo.com

Abstract

This paper explores the research issue and methodology of a query focused multi-document summarizer. Considering its possible application area is Web, the computation is clearly divided into offline and online tasks. At initial preprocessing stage an offline document graph is constructed, where the nodes are basically paragraphs of the documents and edge scores are defined as the correlation measure between the nodes. At query time, given a set of keywords, each node is assigned a query dependent score, the initial graph is expanded and keyword search is performed over the graph to find a spanning tree identifying relevant nodes satisfying the keywords. Paragraph ordering of the output summary is taken care of so that the output looks coherent. Although all the examples, shown in this paper are based on English language, we show that our system is useful in generating query dependent summarization for non- English languages also. We also present the evaluation of the system.

1 Introduction

With the proliferation of information in the Internet, it is becoming very difficult for users to identify the exact information. So many sites are providing same piece of information and a typical query based search in Google results in thousands of links if not million. Web Search engines generally produce query dependent snippets for each result which help users to explore further. An automated query focused multi-document summarizer, which will generate a query based short summary of web pages will be very useful to get a glimpse over the complete story. Automated multi-document summarization has drawn much attention in recent years. Most multi-document summarizers are query independent, which produce majority of information content from multiple documents using much less lengthy text. Each of the systems fall into two different categories: either they are sentence extraction based where they just extract relevant sentences and concatenate them to produce summary or they fuse information from multiple sources to produce a coherent summary.

In this paper, we propose a query focused multi-document summarizer, based on paragraph extraction scheme. Unlike traditional extraction based summarizers which do not take into consideration the inherent structure of the document, our system will add structure to documents in the form of graph. During initial preprocessing, text fragments are identified from the documents which constitute the nodes of the graph. Edges are defined as the correlation measure between nodes of the graph. We define our text fragments as paragraph rather than sentence with the view that generally a paragraph contains more correlated information whereas sentence level extraction might lead to loss of some coherent information.

Since the system produces multi-document summary based on user's query, the response time of the system should be minimal for practical purpose. With this goal, our system takes following steps: First, during preprocessing stage (offline) it performs some query independent tasks like identifying seed summary nodes and constructing graph over them. Then at query time (online), given a set of keywords, it expands the initial graph and performs keyword search over the graph to find a spanning tree identifying relevant nodes (paragraphs) satisfying the keywords. The performance of the system depends much on the identification of the initial query independent nodes (seed nodes). Although, we have presented all the examples in the current discussion for English language only, we argue that our system can be adapted to work in multilingual environment (i.e. Hindi, Bengali, Japanese etc.) with some minor changes in implementation of the system like incorporating language dependent stop word list, stemmer, WodrNet like lexicon etc.

In section 2, related works in this field is presented. In section 3 the overall approach is described. In section 4 query independent preprocessing steps are explained. In section 5 query dependent summary generation and paragraph ordering scheme is presented. Section 6 presents the evaluation scheme of the system. In section 7 we discuss how our system can be modified to work in multilingual scenario. In section 8 we have drawn conclusion and discussed about future work in this field.

2 Related Work

A lot of research work has been done in the domain of multi-document summarization (both query dependent/independent). MEAD (Radev et al., 2004) is centroid based multi-document summarizer which generates summaries using cluster centroids produced by topic detection and tracking system. NeATS (Lin and Hovy, 2002) selects important content using sentence position, term frequency, topic signature and term clustering. XDoX (Hardy et al., 2002) identifies the most salient themes within the document set by passage clustering and then composes an extraction summary, which reflects these main themes.

Graph based methods have been proposed for generating query independent summaries. Websumm (Mani and Bloedorn, 2000) uses a graph-connectivity model to identify salient information. Zhang et al (2004) proposed the methodology of correlated summarization for multiple news articles. In the domain of single document summarization a system for query-specific document summarization has been proposed (Varadarajan and Hristidis, 2006) based on the concept of document graph.

In this paper, the graph based approach has been extended to formulate a framework for generating query dependent summary from related multiple document set describing same event.

3 Graph Based Modeling

The proposed graph based multi-document summarization method consists of following steps: (1) The document set D = $\{d_1, d_2, \ldots\ d_n\}$ is processed to extract text fragments, which are paragraphs in our case as it has been discussed earlier. Here, we assume that the entire document in a particular set are related i.e. they describe the same event. Some document clustering techniques may be adopted to find related documents from a large collection. Document clustering is out of the scope of our current discussion and is itself a research interest. Let for a document d_i, the paragraphs are $\{p_{i1}, p_{i2}, \ldots p_{im}\}$. But the system can be easily modified to work with sentence level extraction. Each text fragment becomes a node of the graph. (2) Next, edges are created between nodes across the document where edge score represents the degree of correlation between inter documents nodes. (3) Seed nodes are extracted which identify the relevant paragraphs within D and a search graph is built offline to reflect the semantic relationship between the nodes. (4) At query time, each node is assigned a query dependent score and the search graph is expanded. (5) A query dependent multi-document summary is generated from the search graph which is nothing but constructing a total minimal spanning tree T (Varadarajan and Hristidis, 2006). For a set of keywords Q = $\{q_1, q_2, ..\ q_n\}$, T is total if $\forall q \in Q$, T consists of at least one node satisfying q and T is minimal if no node can be removed from T while getting the total T.

4 Building Query Independent Components

Mainly there are two criteria for the performance evaluation of such systems: First it's accuracy i.e. the quality of output with respect to specific queries and next of course the turn around time i.e., how fast it can produce the result. Both are very important aspects of such system, and we will show how these aspects are taken care of in our system. Runtime of such system greatly depends on how well the query independent graph is constructed. At one extreme, offline graph can be built connecting all the nodes from each of the documents, constituting a total document graph. But keyword search over such large graph is time consuming and practically not plausible. On the other hand, it is possible to select query specific nodes at runtime and to create a graph over those nodes. But if the number of such nodes is high, then calculating similarity scores between all the nodes will take large computing time, thus resulting in slower performance.

We will take an intermediate approach to attack the problem. It can be safely assumed that significant information for a group of keywords can be found in "relevant/topic paragraphs" of the documents. So, if relevant/topic nodes can be selected from document set D during offline processing, then the significant part of the search graph can be constructed offline which greatly reduce the online processing time. For example, if a user wants to find the information about the IBM Hindi speech recognition system, then the keywords are likely to be {IBM, speech recognition, accuracy}. For a set of news articles about this system, the topic paragraphs, identified offline, naturally satisfy first two keywords and theoretically, they are the most informative paragraphs for those keywords. The last term 'accuracy' (relevant for accuracy of the system) may not be satisfied by seed nodes. So, at run time, the graph needs to be expanded purposefully by including nodes so that the paragraphs, relevant to 'accuracy of the system' are included.

4.1 Identification of Seed/ Topic Nodes

At the preprocessing stage, text is tokenized, stop words are eliminated, and words are stemmed (Porter, 1980). The text in each document is split into paragraphs and each paragraph is represented with a vector of constituent words. If we consider pair of related document, then the inter document graph can be represented as a set of nodes in the form of bipartite graph. The edges connect two nodes corresponding to paragraphs from different documents. The similarity between two nodes is expressed as the edge weight of the bipartite graph. Two nodes are related if they share common words (except stop words) and the degree of relationship can be measured by adapting some traditional IR formula (Varadarajan and Hristidis, 2006).

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

Where $tf(d, w)$ is number of occurrence of w in d, $idf(w)$ is the inverse of the number of documents containing w, and $size(d)$ is the size of the documents in words. The score can be accurately set if stemmer and lexicon are used to match the equivalent words. With the idea of page ranking algorithms, it can be easily observed that a paragraph in a document is relevant if it is highly related to many relevant paragraphs of other document. If some less stringent rules are adopted, then a node from a document is selected as seed/topic node if it has high edge scores with nodes of other document. Actually for a particular node, total edge score is defined as the sum of scores of all out going edges from that node. The nodes with higher total edge scores than some predefined threshold are included as seed nodes. In Figure 1. correlation between two news articles is shown as a bipartite graph.

But the challenge for multi-document summarization is that the information stored in different documents inevitably overlap with each other. So, before inclusion of a particular node (paragraph), it has to be checked whether it is being repeated or not. Two paragraphs are said to be similar if they share for example, 70% words (non stop words) in common.

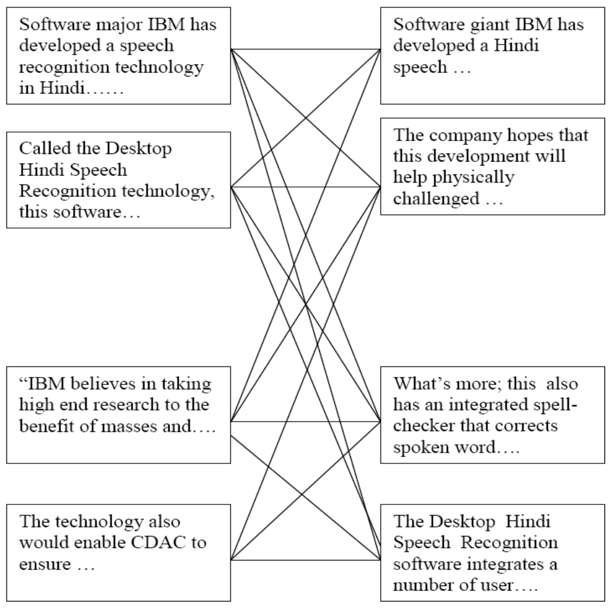

Software major IBM has developed a speech recognition technology in Hindi……
Called the Desktop Hindi Speech Recognition technology, this software…
"IBM believes in taking high end research to the benefit of masses and….
The technology also would enable CDAC to ensure …
Software giant IBM has developed a Hindi speech …
The company hopes that this development will help physically challenged …
What's more; this also has an integrated spell-checker that corrects spoken word….
The Desktop Hindi Speech Recognition software integrates a number of user….

Figure 1. A bipartite graph representing correlation among two news articles on same event.

4.2 Offline Construction of Search Graph

After detection of seed/topic nodes a search graph is constructed. For nodes, pertaining to different documents, edge scores are already calculated, but for intra document nodes, edge scores are calculated in the similar fashion as said earlier. Since, highly dense graph leads to higher search/execution time, only the edges having edge scores well above the threshold value might be considered. The construction of query independent part of the search graph completes the offline processing phase of the system.

5 Building Query Dependent Components

At query time, first, the nodes of the already constructed search graph are given a query dependent score. The score signifies the relevance of the paragraph with respect to given queries. During evaluation if it is found that any keyword is not satisfied by the seed nodes, then system goes back to individual document structure and collects relevant nodes. Finally, it expands the offline graph by adding those nodes, fetched from individual documents. Next, the expanded search graph is processed to find the total minimum spanning tree T over the graph.

5.1 Expanding Search Graph

When query arrives, system evaluates nodes of the offline search graph and computes query dependent score. This computation is based on ranking principals from IR community. The most popular IR ranking is okapi equation (Varadarajan and Hristidis, 2006) which is based on tf-idf principle.

$$\sum_{t \in Q,d} \ln \frac{N - df + 0.5}{df + 0.5} \cdot \frac{(k_1 + 1).tf}{(k_1(1-b) + b\frac{dl}{avdl}) + tf} \cdot \frac{(k_3 + 1).qtf}{k_3 + qtf}$$

tf is the term's frequency in document, qtf is term's frequency in the query, N is the total no. of documents in collection, df is the number of documents that contain the term, dl is the document length (number of words), avdl is the average document length and k1 (1.0 – 2.0), b (0.75), k3 (0 -1000) are constants.

During node score computation, the system intelligently partitions the query set Q into two parts. One part consists of q_i's which are satisfied by at least one node from offline search graph. The other part consists of q_i's which are not satisfied by any node from offline search graph. The system then computes query dependent scores for the nodes of all the individual documents for the unsatisfied keyword set and relevant nodes (having score above threshold) are added to the search graph. Edge scores are computed only for edges connecting newly added nodes with the existing ones and between the new nodes. In this way, the offline graph is expanded by adding some query dependent nodes at runtime. Query dependent scoring can be made faster using a full text indexing which is a mapping $K_i \rightarrow (D_i , N_i)$; where K_i's are content words (i.e., not stop words) and D_i's and Ni's are respectively the document ids and the node ids within the document set. Since, the node score is calculated at runtime, it needs to be accelerated. Thus a full text index developed offline will be of great help.

5.2 Summary Generation

Summary generation is basically a keyword search technique in the expanded search graph. This is to mention that the search technique discussed here is basically based on AND semantic, i.e. it requires all the keywords to be present in the summary, but the algorithm can be modified to take care of OR semantic also. Keyword search in graph structure is itself a research topic and several efficient algorithms are there to solve the problem. DBXplorer (Agrawal et al., 2002), BANKS (Bhalotia et al., 2002), are popular algorithms in this field which consider relational database as graph and devise algorithms for keyword based search in the graph. Finally, Varadarajan and Hristidis (2006) has proposed Top-k Enumeration and MultiResultExpanding search for constructing total minimum spanning tree over a document graph. Any of the above popular algorithms can be adapted to use within our framework.

In our system we have used a search algorithm which finds different combinations of nodes that represent total spanning tree. For each of the combination we compute score of the summary based on some IR principle (Varadarajan and Hristidis, 2006). Then we take the one having best score (minimal in our case). If the graph is not too dense, then the response time will be small enough. The equation given below is used to compute the score of individual spanning tree T.

$$T_{score} = a \sum_{e \in T} \frac{1}{e_{score}} + b \frac{1}{\sum_{n \in T} n_{score}}$$

Where T_{score} the score of the spanning tree, e and n is are edge and node of T respectively, e_{score} and n_{score} are edge score and individual node score respectively. a and b are non zero positive constants in the range of [0 – 1]. For a particular search graph, it is possible to find many total spanning trees, having different summary scores. In our system, the summary with the best score is considered.

In Figure 2 two sample news stories are shown along with system identified seed nodes, shown in bold. A query based summary from that related document set is shown in Figure 3.

5.3 Paragraph Ordering Scheme

In the previous sections, the techniques for generation of summary nodes have been discussed. Here, we will investigate the method for ordering them into a coherent text. In case of single document summarization, sentence/paragraph ordering is done based on the position of extracted paragraphs/ sentences in the original document. But in multi-document scenario, the problem is non trivial since information is extracted from different documents and no single document can provide ordering. Besides, the ordering of information in two different documents may be significantly varying because

P_0: Software giant IBM has developed a speech recognition software in Hindi. P_1 : The company hopes that this development will help physically challenged and less literate Hindi speakers to access information using a variety of applications. P_2 : **The Desktop Hindi Speech Recognition Technology developed by the IBM India Software Lab in collaboration with Centre for Development of Advanced Computing would provide a natural interface for human-computer interaction.** P_3 : The new IBM technology could help to provide a natural interface for human-computer interaction. P_4: According to Dr. Daniel Dias, Director, IBM Indian Research Laboratory, the technology which helps transcribe continuous Hindi speech instantly into text form, could find use in a variety of appli In Figure 1. correlation between two news articles is shown as a bipartite graph. cations like voice-enabled ATMs, car navigation systems, banking, telecom, railways, and airlines. P_5: **Besides, the technology could also enable C-DAC to ensure a high level of accuracy in Hindi translation in a number of domains like administration, finance, agriculture and the small-scale industry.** P_6: The IBM Desktop Hindi Speech Recognition software is capable of recognizing over 75,000 Hindi words with dialectical variations, providing an accuracy of 90 to 95%. P_7: What's more; this software also has an integrated spell-checker that corrects spoken-word errors, enhancing the accuracy to a great extent. P_8: The Desktop Hindi Speech Recognition Technology also integrates a number of user-friendly features such as the facility to convert text to digits and decimals, date and currency format, and into fonts which could be imported to any Windows-based application. P_9: "IBM believes in taking high-end research to the benefit of the masses and bridging the digital divide through a faster diffusion process," concluded Dias.	P_0: **Software major IBM has developed a speech recognition technology in Hindi which would help physically challenged and less literate Hindi speakers access information through a variety of systems.** P_1 : Called the Desktop Hindi Speech Recognition technology, this software was developed by the IBM India Software Lab jointly with the Centre for Development of Advanced Computing. P_2 : **The technology, which helps transcribe continuous Hindi speech instantly into text form, could find use in a variety of applications like voice-enabled ATMs, car navigation systems, banking, telecom, railways and airlines, said Dr Daniel Dias, Director, IBM India Research Laboratory.** P_3 : The system can recognize more than 75,000 Hindi words with dialectical variations, providing an accuracy level of 90-95 per cent, he said. P_4: A spellchecker to correct spoken-word errors also enhances the accuracy of the system. P_5: **The technology also has integrated many user-friendly features such as facility to convert text to digits and decimals, date and currency format, and into fonts which could be imported to any windows-based application.** P_6: "IBM believes in taking high-end research to the benefit of the masses and bridging the digital divide through a faster diffusion process", Dias said. P_7: The technology also would enable C-DAC to ensure high-level accuracy in Hindi translation in a host of domains, including administration, finance, agriculture and small scale industry.

Figure 2. Paragraphs of two news articles with five extracted seed/ topic paragraphs (in bold). Underlined paragraphs are added later during graph expansion phase.

Software major IBM has developed a **speech recognition** technology in Hindi which would help physically challenged and less literate Hindi speakers access information through a variety of systems. [Doc-2, Para - 0]
Besides, the technology could also enable C-DAC to ensure a high level of **accuracy** in Hindi translation in a number of domains like administration, finance, agriculture and the small-scale industry. [Doc-1, Para-5]
A **spellchecker** to correct spoken-word errors also enhances the **accuracy** of the system. [Doc-2, Para - 4]

Figure 3. Automatic summary based on {speech recognition, accuracy, spellchecker} query

the writing styles of different authors are different. In case of news event summarization, chronological ordering is a popular choice which considers the temporal sequence of information pieces, when deciding the ordering process.

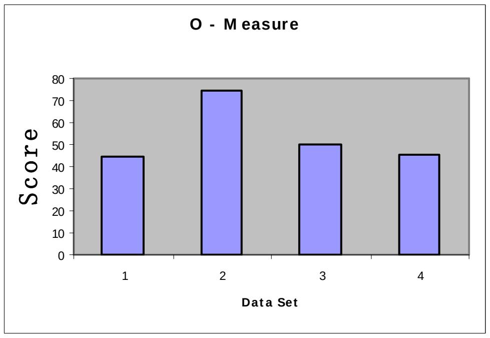
In this paper, we will propose a scheme of ordering which is different from the above two approaches in that, it only takes into consideration the semantic closeness of information pieces (paragraphs) in deciding the ordering among them. First, the starting paragraph is identified which is the paragraph with lowest positional ranking among selected ones over the document set. Next for any source node (paragraph) we find the summary node that is not already selected and have (correlation value) with the source node. This node will be selected as next source node in ordering. This ordering process will continue until the nodes are totally ordered. The above ordering scheme will order the nodes independent of the actual ordering of nodes in the original document, thus eliminating the source bias due to individual writing style of human authors. Moreover, the scheme is logical because we select a paragraph for position p at output summary, based on how coherent it is with the (p-1)th paragraph.

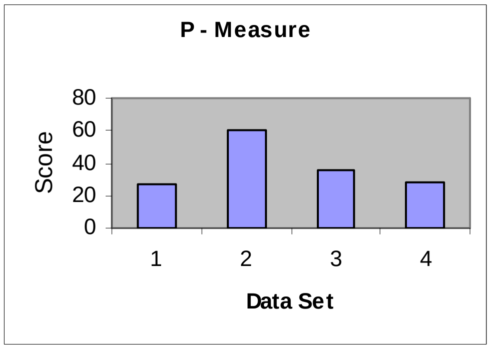
6 Evaluation

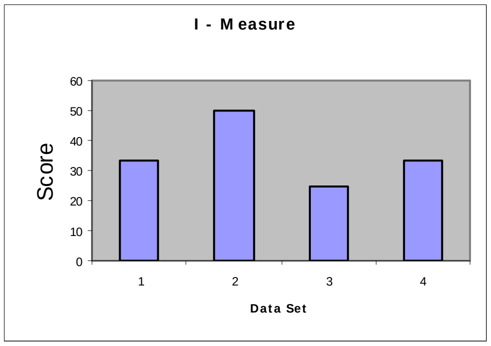
Evaluation of summarization methods is generally performed in two ways. Evaluation measure based on information retrieval task is termed as the *extrinsic* method, while the evaluation based on user judgments is called the *intrinsic* measure. We adopted the latter, since we concentrated more on user's satisfaction. We measure the quality of output based on the percentage of overlap of system generated output with the manual extract. Salton et al (1997) observed that an extract generated by one person is likely to cover 46% of the information that is regarded as most important by another person. Mitra et. al. (1998) proposed an interesting method for evaluation of paragraph based automatic summarization and identified the following four quality-measures – Optimistic (O), Pessimistic (P), Intersection (I) and Union (U) based evaluation. For evaluation purpose, we identify different related document set (D) from different domains like technical, business etc and keyword (query) list for each domain. Users are asked to manually prepare the multi-document summarization based on the given queries. They prepared it by marking relevant paragraphs over D. Based on the excerpts prepared by the users; the above scores are calculated as O: Percentage overlap with that manual extract for which the number of common paragraphs is highest, P: Percentage overlap with that manual extract for which the number of common paragraphs is lowest; I: Percentage overlap with the intersection of manual extracts; U: Percentage overlap with the union of manual extracts. The results are shown in Table 1. A comparative survey of quality measures for the set of articles is shown in Figure 3.

D	$O_{measure}$	$P_{measure}$	$I_{measure}$	$U_{measure}$
article1 & article2	44.4	27	33.3	66.6
article3 & article4	75	60	50	100
article5 & article6	50	35.5	25	66
article7 & article8	45.5	28.7	33.3	56.4

Table 1. Evaluation score







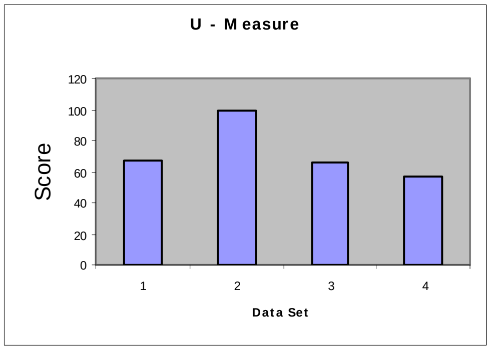


Figure 3. Comparative measure scores for set of articles

7 Baseline Approach to Multilingual Summarization

Our baseline approach to multilingual multidocument summarization is to apply our English based multi-document summarization system to other non-English languages like Hindi, Bengali, Japanese etc. We have initially implemented the system for English language only, but it can be modified to work in multilingual scenario also. To work with other languages, the system requires some language dependent tools for that particular language:

1) A stop word list of that language is required because they have no significance in finding similarity between the paragraphs and need to be removed during initial preprocessing stage.

2) A language dependent stemmer is required. In most of the languages, stemmer is yet to be developed. Another problem is that suffix stripping is not the only solution for all languages because some languages have affix, circumfix etc. in their inflected form. A morphological analyzer to find the root word may be used for those languages.

3) A lexicon for that language is required to match the similar words. For English, WordNet is widely available. For other languages also, similar type of lexicons are required.

If these tools are available then our system can be tuned to generate query dependent multilingual multi-document summary.

8 Conclusion and Future Work

In this work we present a graph based approach for query dependent multi-document summarization system. Considering its possible application in the web document, we clearly divided the computation into two segments. Extraction of seed/topic summary nodes and construction of offline graph is a part of query independent computation. At query time, the precomputed graph is processed to extract the best multi-document summary. We have tested our algorithm with news articles from different domains. The experimental results suggest that our algorithm is effective. Although we experimented with pair of articles, the proposed algorithm can be improved to handle more than two articles simultaneously.

The important aspect of our system is that it can be modified to compute query independent summary which consists of topic nodes, generated during preprocessing stage. The paragraph ordering module can be used to define ordering among those topic paragraphs. Another important aspect is that our system can be tuned to generate summary with custom size specified by users. The spanning tree generation algorithm can be so modified that it produces not only total spanning tree but also takes care of the size requirement of user. Lastly, it is shown that our system can generate summary for other non-English documents also if some language dependent tools are available.

The performance of our algorithm greatly depends on quality of selection of topic nodes. So if we can improve the identification of topic paragraphs and shared topics among multiple documents it would surely enhance the quality of our system.

9 References

A. Singhal , M. Mitra, and C. Buckley. 1997. Automatic Text Summarization by Paragraph Extraction. *Proceedings of* ACL/EACL Workshop.

C.-Y. Lin and E.H. Hovy. 2002. From Single to Multi-document Summarization: A Prototype System and its Evaluation. *Proceedings of ACL:* 457–464.

D.R. Radev, H. Jing, M. Styś and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, Vol.40:919–938.

G. Salton , A. Singhal , M. Mitra, and C. Buckley. 1997. Automatic text structuring and summarization. *Information Processing and Management*: Vol. 33, No. 2: 193-207.

G. Bhalotia, C. Nakhe, A. Hulgeri, S. Chakrabarti and S.Sudarshan. 2002. Keyword Searching and Browsing in Databases using BANKS. *Proceedings of ICDE* : 431-440.

H. Hardy, N. Shimizu, T. Strzalkowski, L. Ting, G. B. Wise and X. Zhang. 2002. Cross-document summarization by concept classification. *Proceedings of SIGIR.02*: 65-69 .

I. Mani and E. Bloedorn. 2000. Summarizing Similarities and Differences Among Related Documents. *Information Retrieval*, Vol. 1(1): 35-67.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

R. Varadarajan,. V. Hristidis. 2006. A system for query-specific document summarization. *Proceedings of CIKM 2006*: 622-631.

S. Agrawal, S. Chaudhuri, and G. Das.2002. DBXplorer: A System for Keyword-Based Search over Relational Databases. *Proceedings of ICDE:* 5-16.

Y. Zhang, X. Ji, C. H. Chu, and H. Zha. 2004. Correlating Summarization of Multisource News with K-Way Graph Biclustering. *SIGKDD Explorations 6*(2): 34-42.