Text Segmentation by Topic

Jay M. Ponte[1] and W. Bruce Croft[1]

Computer Science Department, University of Massachusetts, Amherst, 01002, USA

Abstract. We investigate the problem of text segmentation by topic. Applications for this task include topic tracking of broadcast speech data and topic identification in full-text databases. Researchers have tackled similar problems before but with different goals. This study focuses on data with relatively small segment sizes and for which within-segment sentences have relatively few words in common making the problem challenging. We present a method for segmentation which makes use of a query expansion technique to find common features for the topic segments. Experiments with the technique show that it can be effective.

1 Introduction

There has recently been increased interest in tracking topics in information "feeds." A feed is a continuous stream of text produced, for example, by speech recognition of broadcast news. The text in such an environment contains no mark-up to indicate topic boundaries and may not even delineate sentences. The text also tends to be terse and words are not repeated to the degree they would be in, for example, an academic journal paper. Approaches to text segmentation that depend on redundancy at the word level will fail on such data.

Segmenting text into topics can also be appropriate in heterogeneous full-text database environments. In this case, the topics would be passages in the documents that could be used to improve retrieval or summarization. In this paper, we focus on text segmentation for information feeds and use text for our experiments that reflects some of the characteristics we would expect to encounter. Specifically, the text has a short (2.8 sentence) average topic segment size and within-topic sentences usually do not have enough common words to determine appropriate segmentation. We do, however, assume that sentence boundaries can easily be identified. Figure 1 shows an example of the text from the "What's News" database constructed from the TREC Wall St Journal collection.

The "What's News" articles consist of several topics delimited by groups of dashes. The six sentences in Figure 1 form three topic segments of two sentences each. Notice that, other than stopwords, the first two sentences have only the word "Lebanon" in common, the last two have only the word "Lee" in common and the middle two have no words in common at all. However, within-topic sentences do have many semantically related words and phrases in common. For example, "White House" and "Bush" in the third sentence are related to "presidential spokesman" in the fourth sentence and "cyst" and "cancerous" in

```
Police in Lebanon said that two Red Cross workers abducted last week are
being held by Palestinian guerrillas led by terrorist Abu Nidal.
Meanwhile, thousands of students returned to classes as Lebanon's state
schools, most private schools and the American University of Beirut
reopened after being closed for six months.
---
The White House said that a cyst removed Friday from Bush's right middle
finger wasn't cancerous.
A presidential spokesman said a routine pathological examination was
performed on the half-inch growth following the 25-minute surgical
procedure at Walter Reed Army Medical Hospital.
---
Singapore's Lee Kuan Yew said he would step down as prime minister by the
end of next year, ending three decades of nearly one-man rule in the
island nation.
The 66-year-old Lee, in an interview with the British Broadcasting Corp.,
said he would hand over power to his deputy, Goh Chok Tong.
```

Fig. 1. Example of typical topic segments.

the third sentence are related to "growth" and "surgical procedure" in the fourth sentence. In order to segment this data correctly, we will need to use a method that makes use of the related words. The technique of Local Context Analysis (LCA) allows us to do exactly that. We will revisit this example to show how LCA helps to solve the problem, but we first discuss previous work on similar problems.

2 Previous Work

The topic segmentation task is somewhat related to previous work in passage retrieval. Many passage retrieval techniques have, however, used fixed length passages [1] or other features such as paragraph boundaries [6]. For our task, paragraph and section information is not available and topic segments consist of a very small number of sentences, sometimes only one, so choosing an arbitrary block size as is often done for passage retrieval is not appropriate.

Salton and Singhal [8] and Salton et al [7], discuss the decomposition of text into segments and themes where a segment is contiguous block of text discussing a single subtopic and a theme is a chain of such segments possibly interleaved with other themes. The segmenting process begins at paragraph level. Then, paragraphs are compared by computing cosine similarity in order to infer cohesive multi-paragraph segments.

Hearst [3] and Hearst and Plaunt [4] discuss a method of segmenting expository texts into multi-paragraph subtopics which they call 'tiles' using cosine similarity in conjunction with smoothing. One notable feature of this work is that formatting information is not used in the segmentation process. Instead, the text is initially broken up into into blocks of size N with N ranging from three to five sentences in the experiments. Next, a similarity curve of adjacent blocks is computed using cosine similarity. This curve is then smoothed to get rid of local extrema. Finally, the resulting smoothed graph is used to identify potential topic boundaries. Hearst and Plaunt [4] performed experiments using these topic boundaries to enhance ad hoc retrieval, but the results were not significantly better than fixed-sized windows. In [3] the segmentation was compared to that of several human judges.

These approaches are somewhat different from the one taken in our preliminary study. In the first place, we cannot assume paragraph boundaries are available. Secondly, we cannot use a fixed block size greater than one sentence since, in our data, topic segments can be very short, sometimes only one or two sentences. Finally, for the data in our study, we cannot assume that within-topic sentences have very many, or indeed any, words in common. In that case, measuring similarity of within topic sentences does not provide enough information.

As mentioned earlier, much of the work on passage retrieval has used passages of fixed length. An exception to this is Mittendorf and Shäuble [5] in which Hidden Markov Models (HMMs) were used to retrieve relevant passages of variable length. This is an interesting approach and is somewhat related to our work in that in both cases the text is broken up using a sequential (Markov) decision process. The difference is that in their approach, a specific information need is modeled by a stochastic process which generates text fragments relevant to a particular query. This process is called the passage model. A second process generates typical text fragments without regard to any query. This process is called the background model. A text block is scored as a function of the probability that the passage model produced the block and the background model produced the surrounding context of the block. Our approach, on the other hand, is to segment the text by topic independent of a specific query and so we do not explicitly model an information need in the current study.

3 Methodology

We now present a brief overview of the method, followed by additional detail on key aspects. As mentioned earlier, one of the distinguishing features of this problem is that within topic sentences often do not share any common words. However, as shown in Figure 1, semantically related words will be present in a topic segment.

We considered the sentence to be the smallest unit, i.e. a segment consists of one or more sentences. Note that in broadcast speech data, sentence boundary identification is not a trivial problem. For the current data, it can be approximated reasonably well.

As a first step, we used the method of Local Context Analysis (LCA), as described by Xu and Croft [9], in order to find words and phrases related to each sentence. The words and phrases returned by LCA were used in place of the original sentence and pairwise similarity of all sentences in the data set was calculated. Actually, sentences were only compared within a fixed window to save unnecessary computation, but conceptually, that is not important.

The pairwise similarity measures were then used to score individual segments of various sizes. Specifically, for n ranging from one to the maximum segment size, each block of size n at each position was assigned a score based on the pairwise similarity measures. These scores were used to rank each potential block of size n starting at each position in the text.

Finally, the segmentation was done using dynamic programming. Each block was given a final score based on its rank position and length. Each possible segmentation, i.e. each possible sequence of topic breaks, was considered to find the one that maximized the total score. There are exponentially many possible segmentations, but dynamic programming makes the calculation tractable.

Now that we have seen an overview of the process, we will look at some of the details.

3.1 LCA

Xu and Croft [9] developed Local Context Analysis as the method of query expansion somewhat like a more robust version of local feedback (see [2] for a discussion of local feedback). For the purposes of the current work, LCA is being used as an association thesaurus. Each sentence in the test set is posed as a query to the LCA database.

LCA works as follows, given a query:

– Retrieve the top N passages.
– Extract words and phrases from these passages using a variant of EMIM.
– Rank the extracted concepts according to their co-occurrence with the query terms and their collection statistics.
– Return the top M concepts.

LCA is quite robust with respect to the choice of N, the number of passages to use and M, the number of top concepts to add. For the purposes of this study, N was 2000 and M was 100.

The concepts returned from the LCA database are strongly associated with the original sentences. We believe the reason for the effectiveness of LCA for this task is that the words in context tend to disambiguate each other and LCA is a method that takes advantage of that fact since it favors co-occurrence with multiple query terms, or, in this case, multiple words in the sentence.

Recall the six-sentence example in Figure 1, after running LCA, we end up with a set of surrogate features for each sentence.

Table 1 summarizes the number of co-occurring concepts after LCA expansion.

Recall from Figure 1 that sentences 1 and 2 form a single topic and where as the original sentences had only one word in common, after LCA expansion, their surrogates have ten features in common. Likewise, sentences 3 and 4 formerly had no words in common, now they have eleven features in common and finally

Sentence	1	2	3	4	5	6
1	—	10	0	1	0	0
2	10	—	0	0	1	3
3	0	0	—	11	1	1
4	1	0	11	—	0	0
5	0	1	1	0	—	65
6	0	3	1	0	65	—

Table 1. Number of common LCA features for six example sentences.

sentences 5 and 6 which had one word in common, now have sixty five features in common. Equally important, pairs of across-topic sentences have few concepts in common.

3.2 Pairwise Sentence Similarity

After LCA expansion, we use the LCA concepts to measure the similarity of blocks of text. As a first step, each pair of sentences, is assigned a similarity score based on coordination level matching on the list of LCA concepts. The similarity function is simply the sum of the number of matches. For example for sentences 1 and 2 in Figure 1, the similarity score will be 10. This is a very simple measure. We assume that the top 100 concepts from LCA are all going to be useful and so we do not use a more elaborate similarity measure. In order to avoid unnecessary computation, only sentences within a fixed window were assigned a score. The window size depends on the calculations for each block, which will be covered shortly, and the range of possible block sizes.

3.3 Segment Ranking

For each possible segment size, we want to compute a score for each position in the input text. We use sizes ranging from 1 to N where N is a conservative estimate of the maximum segment size. The features we will use are *internal similarity* which is simply the sum of the pairwise similarities within the segment and *left and right external similarity. Left external similarity* is the sum of the pairwise similarities of each sentence in the segment to a fixed number of preceding sentences. Similarly, *right external similarity* is the sum of the pairwise similarities of each sentence in the segment to a fixed number of sentences following the segment. For example, with a window of size two, sentences 3 and 4 in Figure 1 will have an *internal similarity* score of 11, a *Left external similarity* of 1 and a *right external similarity* of 2. For each size, the segments will be ranked by the *internal similarity* minus the two *external similarities.*

3.4 Dynamic Programming

We now have ranking for segments of size N at each position in the input text i.e. for each block size N a score is computed for the block from p to $p + n$ for each position p in the input stream. The rank position of each segment will be used to indicate the belief that a segment at a given position is of length N. At this point we use dynamic programming to consider every possible segmentation.

As part of the dynamic programming process, we use a Gaussian length model to weight each potential segment with the prior probability of the segment length. The length model is defined as:

$$\frac{k}{\sqrt{2\pi}\sigma} e^{-(\frac{1}{2})(\frac{(x-\mu)}{\sigma})^2}$$

Where μ is the estimated mean topic length, σ is the estimated standard deviation and k is constant for scaling purposes. The parameters were estimated from a training set consisting of 85 topic segments. As will be shown, while the length model improves performance, it is not necessary for reasonable segmentation.

4 Experiments

Experiments were run on three data sets. Also, the method with and without LCA was compared on one of the data sets. In addition, the method with and without length modeling was compared on one of the data sets. Finally, an experiment was run to study the time sensitivity of LCA for this task.

4.1 Test Data

The test data consisted of three sets drawn at random from the "What's News" articles from Wall Street Journal 1989. These articles are a good choice for evaluating segmentation in the two contexts mentioned previously. In order to segment stories in a topic tracking environment one needs to identify boundaries in relatively terse text in which formatting information such as paragraph or section boundaries are not available. Likewise, in a heterogeneous text collection, one may have documents which really contain a number of unrelated topics. In this case, one needs to predict the boundaries of these unrelated document portions, furthermore, as in the topic tracking task, one may not have consistent formatting information available. Since this is the case for these two tasks, the "What's News" articles provide a reasonable set of test data. The form of these articles does not contain information when the topic boundaries have been removed and the terseness and relatively short segment length of these articles makes the task challenging since there is minimal overlapping of content words and minimal structure to the individual stories themselves.

Test set 1 consists of 228 sentences and 86 topics, set 2 consists of 251 sentences and 96 topics and set 3 consists of 269 sentences and 119 topics. An additional set of similar size was used to obtain the parameters for the Gaussian length model. The parameters, $\mu = 2.7$ and $\sigma = 1.4$ are not optimal for the test sets since each set has a different length distribution. Nevertheless, the length model with these parameters worked well, as we shall see shortly.

4.2 Evaluation

We make the assumption that the topic breaks in the original Wall Street Journal "What's News" articles are the ground truth for evaluation purposes. This is a reasonable assumption for this study since each topic segment is a discussion of a different event. In other cases, one might have to determine the level of agreement between several human judges in order to estimate performance, see [3].

In order to score the segmentation generated by the algorithm, we first perform a least squares alignment with the correct segmentation. Then the distance between the two is measurable in terms of *insertions*, *deletions* and *moves*. An *insertion* error is the algorithm produced a break that does not line up with a real break. A *deletion* error is when a real break exists but the algorithm did not produce a break that lines up with it. Finally, a *move* error is when two breaks line up but are not in the same place. Table 2 shows an example. The first column shows the true segmentation and the second column shows the predicted segmentation. In this example, an *insertion* error occurs at sentence 5 where the algorithm has predicted a break which does line up with a real break. A *deletion* error occurs at sentence 27 where the original segmentation has a break which is not found by the algorithm. Finally, two *move* errors occur at sentences 26 and 31 where the algorithm has predicted breaks when the real breaks are at sentences 25 and 29.

Actual Segmentation	Predicted Segmentation
2	2
	5
10	10
14	14
19	19
25	26
27	
29	31
35	35

Table 2. Example of aligned segmentations.

Given the aligned segmentations one can choose an error function for the *move* errors based on the desired level of tolerance. An *insertion* or *deletion* always counts as one error. We report results using two error functions. The first is a 'pessimistic' error function:

$$f(b) = \begin{cases} 1 & \text{if } \exists r(b = r); \\ 0 & \text{otherwise} \end{cases}$$

This function simply counts each break b if and only if it matches some real break r. This measure does not take into account predictions that were close but not exact, e.g. a block of length six would count as a complete miss even if the real block started in the same place and was of length seven. This exact match error function does not distinguish between some segmentations where it clearly should. For example, suppose a data set consists of one hundred sentences with a topic break at sentence fifty. The exact match error function would not distinguish between an algorithm that predicted a single break at sentence forty-nine and one that predicted a single break at sentence two even though the former is clearly better. However, the following error function does make that distinction:

$$f(b) = \left(\frac{1.0}{2d}\right) \times \left(\frac{((u-g)-d)}{u}\right)$$

In this function, d is the difference between the predicted break b and the corresponding real break (where correspondence is determined by the alignment process), u is the difference between the next actual break and the previous one and g is the number of insertion errors between the next actual break and the previous one. It may help to think of u as the amount of uncertainty and g as the number of guesses. This partial match error function gives full credit for exact matches and partial credit for near misses. The first term causes the amount of credit to drop off as the distance increases and the second term measures the 'nearness' of the near miss.

For the purposes of this study, we report scores with both of the above error functions. The errors are counted using both error functions and then the counts are used to calculate recall and precision scores for each function. Recall is measured as the percentage of topic breaks in the original data that were predicted by the algorithm. Precision is measured as the percentage of breaks that were predicted by the algorithm that appeared in the original data.

The two sets of recall and precision scores allow more meaningful comparisons than either set by itself. Two segmentations can be compared by the worst case analysis using the exact match score. The partial match score provides a reality check if the exact match scores are close. Also, for a single segmentation, the partial match scores can be compared to the exact match scores to determine the closeness of missed breaks.

4.3 Results

Set 1	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
Original Words	70.0%	62.9%	77.9 %	70.1%
LCA Concepts	88.8%	82.6%	91.4%	85.1%

Table 3. Comparison with length modeling, with and without LCA expansion.

Table 3 shows the results on Set 1 using the LCA concepts vs. using the original words. It can bee seen that performance with the original words is poor, especially the exact match precision. This is not surprising in light of the fact that many within-topic sentences do not have many words in common. There is simply not enough information in the original sentences to perform the segmentation well. On the other hand, when the LCA concepts are used, the results improve dramatically.

Set 1	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
No Length Model	73.5%	76.3%	75.2%	78.1%
Length Model	88.8%	82.6%	91.4%	85.1%

Table 4. Comparison using LCA expansion with and without length modeling.

Table 4 shows the results on set 1 with and without length modeling. As you can see, without length modeling, performance is not as good but is still reasonable. This is encouraging since one may not be able to estimate an accurate length model in general settings. On the other hand, if a length model is available, segments of typical length have higher probability and a better segmentation results.

Set	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
1	88.8%	82.6%	91.4%	85.1%
2	78.4%	79.2%	79.9%	80.7%
3	75.0%	85.7%	76.6%	87.6%

Table 5. Results for three data sets using LCA and length modeling.

Table 5 shows the results on all three data sets using length information and LCA. These results show that the method is reasonably robust across the three data sets. For purposes of comparison, results using the original words for data sets 2 and 3 are included in in Table 6, notice that the improvement provided by LCA is consistent across the three data sets.

Set	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
2	58.8%	61.3%	66.8%	69.7%
3	58.3%	70.7%	64.6%	78.3%

Table 6. Results for sets 2 and 3 using the original words.

4.4 Time Sensitivity of LCA

The following is an example where the method fails. The six sentences in Figure 2 form three topics of two sentences each. The correct topic boundaries are represented by the dashes. The segmentation algorithm broke this passage in the middle forming two topics of three sentences each, represented in Figure 2 by the asterisks. One of the reasons for the error is that there is more overlap in the LCA concepts for the across topic sentences. For example, sentence three has concepts such as "parliament", "party" and "election" in common with sentences one and two. Many of the concepts returned for sentences three and four are related to political unrest and reform, probably from editorial articles. For example, sentence four and sentence six have "Mandela" in common.

```
Britain's Conservative Party prepared to open its annual conference today
with its popularity at an eight-year low, the economy in trouble, and
polls showing voters increasingly disenchanted with Prime Minister
Thatcher's recent programs.
The four-day forum in Blackpool, England, follows a gathering last week
by the Labor Party.
---
Yugoslavia's Premier Markovic traveled to the U.S., where he is expected
to seek $1 billion in assistance to bolster his economic and political
restructuring plans.
***
Markovic is to meet with Bush and other administration officials as well
as with commercial bankers during his six-day visit.
---
South Africa said escalating violence between rival political groups in
Namibia could jeopardize a U.N. independence plan for the territory.
U.N. officials are expected to urged Namibia's political leaders to halt
the "mob behavior" by supporters, which resulted in two deaths over the
weekend.
```

Fig. 2. Example of text which caused a problem.

The passage in Figure 2 is from Wall Street Journal, 1989. The original results used Tipster volume 2 and 3 (Tip23) as the LCA database. Tip23 contains articles from Wall Street Journal for the years 1990, 1991 and 1992 (and articles from several other sources). Since these passages are (were) current events, the obvious question is whether an LCA database from a closer time period would improve results since, of late, one is less likely to find articles about Yugoslavian Premier Markovic.

We ran an additional experiment using Tipster volume 1 (Tip1) which includes Wall Street Journal for the years 1987, 1988 and 1989. This is a reasonable approximation of the standard routing task since one would typically keep a collection of articles from the past up to the present time. The above problem goes away under these circumstances and the overall results improve as shown in table 7.

Set 1	Exact Match Recall	Exact Match Precision	Partial Match Recall	Partial Match Precision
Tip23 LCA	88.8%	82.6%	91.4%	85.1%
Tip1 LCA	95.0%	84.4%	95.9%	85.2%

Table 7. Results with more timely LCA database vs. original LCA database.

Note that Tipster volume 1 includes the data in the above example. This does not compromise the result since LCA is not taking advantage of the topic boundaries. One could break up incoming text into large blocks, apply LCA and accomplish essentially the same thing. In fact, the LCA database was built with a passage size of 300 words. That being the case, one might expect this database to cause problems since LCA could potentially return concepts from adjoining topics thereby making them seem more similar. However, LCA is quite robust in this respect and so that was not the case.

5 Performance

An important factor for topic segmentation is the performance of each component of the system. Remember, the four components are LCA expansion, computation of pairwise sentence similarity, segment ranking and dynamic programming. All timing figures are for test set 1 on a DEC AlphaStation 250.

The LCA expansion requires one query per sentence. Assuming 2 seconds per query, on average, this translates into approximately 200 KB per hour for the LCA expansion. Clearly, it is important to speed this up for many applications.

Pairwise sentence similarity is $O(mn^2)$, where m is the number of LCA concepts and n is the number of sentences. However, m can be regarded as a constant and in practice one can reduce this to $O(n)$ by using a fixed sized window with a constant factor for the window size. Pairwise similarity computation takes approximately 3 seconds for set 1, though this could certainly be sped up as the current implementation was built for flexibility rather than speed.

Segment ranking is $O(mn)$ where m is the maximum segment size and n is the number of sentences. In practice, m may be regarded as a constant. Segment ranking takes approximately 0.05 seconds for set 1. Similarly, the dynamic programming step is $O(n)$ with a constant factor for maximum window size. The dynamic programming step takes approximately 0.06 seconds for set 1.

The important point is that the last three steps are linear in the size of the data, all with reasonable constants. The LCA expansion is the expensive part.

6 Conclusions and Future Work

We have presented a method for segmenting text by topic that works well in spite of small segments with few common words. We have shown that LCA is a good technique to augment short passages with related words for the purposes of text segmentation and that matching the age of the LCA database to the age of the data to be segmented is helpful but not critical. In addition, we have shown that segment length modeling is an important component of this task.

The current results suggest some additional experiments. LCA works with a fixed passage size. In the current work, we did not attempt to find the optimal passage size or to study the effects of varying the passage size. It seems reasonable that a passage size related to the mean segment length would improve performance but this remains to be tested.

An additional parameter is the number of concepts to use in place of the original sentence. In the current experiments, we used the top 100 concepts from LCA. In the future we intend to investigate the effects of varying this number.

The length model used in the current method worked well. Larger, more heterogeneous collections may require more complex models, and it may be the case that length modeling will not be feasible. Fortunately, performance without a length model is quite reasonable. We intend to investigate whether that will still be the case in other collections.

In order to handle automatically recognized speech data, we need to extend the method to work in the case where sentence boundaries are difficult, if not impossible, to identify accurately. Other cues may be available, such as length of pauses and it may be possible to use this information instead of sentence boundaries. Obviously, pauses will not generally coincide with sentence boundaries, so, at best, use of this information will present a challenge. Other approaches such as a fixed window may also work well. This remains a matter for investigation.

Finally, the performance issue will need to be addressed. We would like to speed up the current method without resorting to more machine power. We intend to investigate whether LCA can be customized for this task in a more time efficient manner.

7 Acknowledgments

The authors would like to thank Jonathan Yamron of Dragon Systems Inc. for his help with the evaluation metrics.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

1. Callan J. P., "Passage-Level Evidence in Document Retrieval." In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, July, 1994 (pp. 302-310).
2. Croft, W. B. and D. J. Harper. "Using probabilistic models of document retrieval without relevance information." Journal of Documentation, 35, 1979 (pp. 285-295).
3. Hearst, M. "Multi-Paragraph Segmentation of Expository Text", Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, NM, June 1994.
4. Hearst, M. and Plaunt, C. Subtopic Structuring for Full-Length Document Access, Proceedings of the sixteenth Annual International ACM/SIGIR Conference, Pittsburgh, PA. 1993 (pp. 59-68).
5. Mittendorf E. and P. Shäuble, "Document and Passage Retrieval Based on Hidden Markov Models", In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, July, 1994 (pp. 318-327).
6. Salton, Gerard, J. Allan and C. Buckley, "Approaches to Passage Retrieval in Full Text Information Systems", Proceedings of the sixteenth Annual International ACM/SIGIR Conference, Pittsburgh, PA. 1993 (pp. 49-58).
7. Salton,Gerard, Amit Singhal, Chris Buckley and Mandar Mitra. "Automatic Text Decomposition Using Text Segments and Text Themes", Proceedings of the Seventh ACM Conference on Hypertext, Washington D.C., 1996.
8. Salton,Gerard and Amit Singhal. "Automatic Text Theme Generation and the Analysis of Text Structure", Cornell Computer Science Technical Report 94-1438, July 1994.
9. Xu, Jinxi and W. Bruce Croft, "Query Expansion Using Local and Global Document Analysis", In Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, August, 1996 (pp. 4-11).