QueSTS: A Query Specific Text Summarization System

M Sravanthi, C R Chowdary and **P Sreenivasa Kumar**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai, India 600 036.
{sravanti,chowdary and psk}@cse.iitm.ac.in

Abstract

Effective extraction of query relevant information present within documents on the web is a nontrivial task. In this paper we present our system called QueSTS, which does the above task by filtering and aggregating important query relevant sentences distributed across a set of documents. Our approach captures the contextual relationships among sentences of all input documents and represents them as an "integrated graph". These relationships are exploited and several subgraphs of integrated graph which consist of sentences that are highly relevant to the query and that are highly related to each other are constructed. These subgraphs are ranked by our scoring model. The highest ranked subgraph which is rich in query relevant information and also has sentences that are highly coherent is returned as a query specific summary.

Introduction

Huge amounts of information is being added to the World Wide Web (WWW) continuously. So, information overload has become a problem. Information Retrieval (IR) systems such as Google, Yahoo etc. address the problem of information overload by identifying documents relevant to the user's query, ranking them and presenting them as an ordered list. But the number of search results is very high and information pertaining to a query might be distributed across several sources. So it is a tedious task for a user to sift through the search results and find the information she needs. It would be very useful to have a system which could filter and aggregate information relevant to the user's query from various sources and present it as a *digest or summary*. This summary would help in getting an overall understanding of the query topic. Our query specific text summarization system (QueSTS) fulfills this objective by generating a summary that is specific to the given query on a set of related documents.

The query biased summarization of general purpose articles available on web poses significant challenges like maintaining coherence, intelligibility and non-redundancy. *Coherence* determines the readability and information flow, while *intelligibility/responsiveness* is the property that determines if the summary satisfies user's needs or not. In this paper, we present a novel multidocument summarization approach that generates a query specific summary that is concise, coherent, intelligible and complete (covering all query terms). Each document in the input is represented as a *Contextual Graph (CG)* and *CG's* of all documents in the input are merged incrementally to form an *Integrated Graph (IG)*. Each sentence in the documents is represented by a node and contextual similarity between sentences is represented by an edge. Integrated Graph reflects inter and intra document contextual similarities present between sentences of the input. We use "node" and "sentence" interchangeably hereafter.

In these graphs, neighbourhood of a node represents a *context*. Each edge is given a contextual similarity weight and a node is given a relevance weight w.r.t each query term in IG. Based on these weights we explore neighbourhood of all nodes in *IG* to construct a contextual tree($CTree$) and a summary graph($SGraph$). The $CTree$ contains information relevant to a particular query term and a $SGraph$ contains information relevant to the whole query. A $SGraph$ rich in information(high node weights) and coherence (high edge weights) is given as a query specific summary.

Our approach is domain independent and doesn't use any linguistic processing, making it a flexible and general approach that can be applied to unrestricted set of articles found on WWW. Experimental results prove the strength of our system.

Related Work

We focus on extraction based query specific summarization approach. Extraction based approaches use a scoring function which considers statistical and linguistic features like term frequency, cue words, title, position(Luhn 1958; Edmundson 1969; Hovy & Lin 1997), lexical chains(Barzilay & Elhadad 1997) etc. and assigns each sentence a salience score. Several clustering based approaches(Radev, Jing, & Budzikowska 2000) were tried where similar sentences are clustered and a representative sentence of each cluster is chosen as a digest. MEAD(Radev, Jing, &

Budzikowska 2000) is a centroid based multidocument summarizer. It uses features like cluster centroids, position etc., to summarize documents. The documents are clustered together apriori by a topic detection system. We have used *query specific* version of MEAD for comparision in our evaluations. Some other machine learning approaches other than clustering have also been tried out in (Chuang & Yang 2000; John M. Conroy & Stewart 2005; Fisher *et al.* 2005).

Recently, graph based models are being used to represent text. They use measures like degree centrality (Salton *et al.* 1997) and eigen vector centrality (Erkan & Radev 2004; Mihalcea & Tarau 2004) to rank sentences. Inspired by PageRank these methods build a network of sentences and then determine the importance of each sentence based on its connectivity with other sentences. Highly ranked sentences form a summary. Wan et.al. (Wan, Yang, & Xiao 2007) does topic focused multidocument summarization by exploiting relationships between sentences based on manifold ranking process. Bosma et.al. (Bosma 2006) generates a query specific coherent summary by including supporting sentences and uses an entailment algorithm to avoid redundancy.

Efforts reported in DUC[1] also address query specific multidocument summarization. The queries considered there are long questions, typically having 20 to 30 words. But in the context of a search engine, the keywords given are few. Hence we do not compare our system with DUC efforts. We propose a novel algorithm that does statistical processing to exploit the dependencies between sentences and generates a summary by balancing both coherence and query responsiveness in it.

QueSTS Framework

Contextual Graph

We use a graph model to appropriately represent the inherent structure present in the document. The documents are initially preprocessed to remove non-textual elements like images, tables etc. Then the text is segmented using delimiter based approach. We use "." as a delimiter for our experiments and documents are segmented into sentences. These set of sentences of each document are represented as a graph where related sentences are connected to each other. Therefore each document is represented as a *contextual graph* which is defined as:

Definition 1 ***Contextual Graph(CG)****: A contextual graph for a document d is defined as a weighted graph, $CG(d) = (V(d), E(d))$ where $V(d)$ is a finite set of nodes where each node is a sentence in the document d and $E(d)$ is a finite set of edges where an edge $e_{ij} \in E(d)$ is incident on nodes $i, j \in V(d)$ and is assigned a weight reflecting the degree of similarity between nodes i and j. An edge exists if its weight exceeds a threshold μ. Edges connecting adjacent sentences in a document are retained, irrespective of the threshold.*

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

An edge weight $w(e_{ij})$ quantifies contextual similarity between sentences s_i and s_j. It is computed using cosine similarity measure($sim(\overrightarrow{s_i}, \overrightarrow{s_j})$). The weight of each term t is calculated using $tf_t * isf_t$ where tf_t is the term frequency and isf_t is inverse sentential frequency i.e., $log(\frac{n}{n_t})$ where n is the total number of sentences and n_t is number of sentences containing the term t in the graph under consideration. Stop words are removed and remaining words are stemmed before computing these weights. A low similarity value reflects a weak context (i.e. sentences that are not related) and such edges with an edge weight below a threshold μ(=0.001) are discarded. These edges and their weights reflect the degree of coherence in the summary.

Integrated Graph

For multidocument summarization, the input is a set of documents related to a topic. Let $\chi = (D, T)$ be a set of documents D on a topic T. Contextual graphs for each document in this set χ are merged incrementally to form a graph called as *integrated graph*. As the documents in the input set χ are related to a topic T, they will be similar and may contain some sentences which are redundant. These redundant sentences are identified and only one of them is placed in the integrated graph. To reflect the similarities between documents, we establish edges across nodes of different documents and the edge weights of IG are computed. Thus the integrated graph reflects inter as well as intra document similarity relationships present in document set χ. Algorithm for integrated graph construction is given in later sections.

When a query related to the topic T is given, we compute relevancy scores of each node w.r.t each query term. During this process, we have to deal with sentences that are not related to query directly (by having terms), but are relevant. We call these as *supporting sentences*. To handle this we compute centrality based query biased sentence salience weights that not only consider the local information i.e., whether the node contains the query terms, but also global information like the nodes to which it is connected to. We use a mixture model proposed by Otterbacher et.al. in (Otterbacher, Erkan, & Radev 2005) to define importance of a node w.r.t a query term in two aspects: the relevance of sentence to the query term and the kind of neighbours it is connected to. Initially each node is initialized to query similarity weight and then these weights are spread to their neighbours via the weighted graph IG. This process is iterated until the weights converge to a steady state. Adapting this idea, the node weights for each node w.r.t each query term $q_i \in Q$ where $Q = \{q_1, q_2, ..., q_t\}$ are computed using the following equation.

$$w_q(s) = d\frac{sim(s,q)}{\sum_{m \in N} sim(m,q)} + (1-d) \sum_{v \in adj(s)} \frac{sim(s,v)}{\sum_{u \in adj(v)} sim(u,v)} w_q(v) \quad (1)$$

where $w_q(s)$ is node weight of node s with respect to query term q, d is bias factor, N is number of nodes and

$sim(s_i, s_j)$ is cosine similarity between sentences s_i and s_j. First part of equation computes relevancy of nodes to the query and second part considers neighbours' node weights. The bias factor d gives trade-off between these two parts and is determined empirically. For higher values of d, more importance is given to the similarity of node to the query when compared to the similarity between neighbours. The denominators in both terms are for normalization. When a query Q is given to the system, each word is assigned weight based on $tf * isf$ metric and node weights for each node w.r.t each query term are calculated.

Intuitively a node will have a high score value if: 1) it has information relevant to the query and 2) it is connected to similar context nodes which share query relevant information. If a node doesn't have any query term but is linked to nodes having it, then it's neighbour weights are propagated in proportion to the edge weight such that it gets a weight greater than zero. Thus high node weight indicates a highly relevant node present in a highly relevant context and is used to indicate the richness of query specific information in the node.

CTree and SGraph

For each query word, the neighbourhood of each node in the graph IG is explored and a tree rooted at each node is constructed from the explored graph. These trees are called as contextual trees($CTrees$) and they contain nodes and edges that are prominent in the context. A $CTree$ is formally defined as follows.

Definition 2 ***Contextual Tree(CTree)****: A $CTree_i = (N_i, E_i, r, q_i)$ is defined as a quadruple where N_i and E_i are set of nodes and edges respectively. q_i is i^{th} term in the query. It is rooted at r with atleast one of the nodes having the query term q_i. Number of children for each node is atmost b(beam width). It has atmost $(1 + bd)$ nodes where d is the maximum depth. $CTree$ is empty if there is no node with query term q_i within depth d.*

$CTrees$ corresponding to each query term that are rooted at a particular node, are merged to form a summary graph($SGraph$) which is defined as follows:

Definition 3 ***Summary Graph(SGraph)****. For each node r in IG, if there are t query terms, we construct a summary graph $SGraph = (N', E', Q)$ where N' and E' are union of the set of nodes and edges of $CTree_i$ rooted at r respectively and $Q = \{q_1, q_2, ..., q_t\}$.*

Scoring Model

$CTrees$ formed from each node in IG are assigned a score that reflects the degree of coherence and information richness in the tree. It is defined as:

Definition 4 ***CTreeScore****: Given an integrated graph IG and a query term q, score of the $CTree_q$ rooted at node r is calculated as*

$$CTreeScore_q =$$

$$\beta w_q(r) \; + \sum_{\substack{(u,v) \in CTree_q \\ u \; is \; parent \; of \; v}} [\frac{\alpha w(e_{u,v}) + \beta w_q(v)}{(\sqrt{level(v)})}] \quad (2)$$

$$\alpha = \frac{a}{b} * 1.5 \quad (3)$$

where a is average of top three node weights among the neighbours of u excluding parent of u and b is maximum edge weight among nodes incident on u.

The $SGraphs$ formed from each node by merging $CTrees$ for all query terms are ranked using following equation and the highest ranked graph is retained as summary.

Definition 5 ***SGraphScore****. Given an integrated graph IG and a query $Q = \{q_1, q_2, ..., q_t\}$, score of the $SGraph$ SG is calculated as*

$$SGraphScore = \frac{1}{\sqrt{size(SG)}} \sum_{q \in Q} CTreeScore_q \quad (4)$$

The function $size(SG)$ is defined as number of nodes in $SGraph$. Using both edge weights representing contextual similarity and node weights representing query relevance for selecting a node connected to root node, has never been tried before. Our summary graph construction is a novel approach which tries to effectively balance both coherence and informativeness in a summary.

QueSTS Summarization Methodology

Based on the data model and scoring model presented in the above section, we design efficient algorithms to automatically generate query biased summaries from text.

Integrated Graph Construction

Integrated graph represents the relationships present among sentences of the input set. As a long document(the document with maximum number of sentences) tends to contain more number of subtopics than others it's CG is chosen as a *base graph* and is added to IG which is empty initially. The documents in the input set are ordered in decreasing order of their size(number of sentences) and $CG's$ of each document in the ordered list is added incrementally to IG (nodes in a CG are considered in their respective document order).

There are two important issues that need to be addressed in multidocument summarization. They are redundant sentences and ordering of sentences in summary. Redundant sentences are identified as those sentences which have similarity exceeding a threshold λ. This similarity is computed using cosine similarity and experimentally it was found that

$\lambda = 0.7$ is sufficient in most of the cases. During the construction of IG, if the sentence in consideration is found to be highly similar to any sentence of a document other than document being considered in IG, then it is discarded. Otherwise it is added as a new node and is connected to existing nodes with which its similarity is above the threshold μ. Sentence ordering in summary affects readability. For this purpose, we follow a *encoding strategy* and assign an "id" to each node in IG such that there is information flow in summary when nodes are put in increasing order of their ids.

Encoding Strategy: Initially all nodes in the base graph are assigned ids as follows. The i^{th} sentence is assigned $(i-1) * \eta$ as id[2]. This interval η is used to insert all the nodes from other documents that are closer to i (i.e. the inserted node has maximum edge weight with i among all nodes adjacent to it). The sentences in an interval are ordered in decreasing order of their edge weights with i. When a new node is added to IG, an id is assigned to it. Pseudo code for IG construction is given in Algorithm 1.

Algorithm 1 Integrated Graph Construction

```
1: Input: Contextual Graphs CG_i in the decreasing order of number of nodes
2: Output: Integrated graph IG
3: Integrated Graph IG = CG_0 {//base graph}
4: Set id of each node in IG as described in IG Construction
5: i = 1
6: while i ≤ number of CG's do
7:    for each node n ∈ CG_i considered in the document order do
8:       if parent(n) precedes n in the i^th document then {//parent is the maximum weighted adjacent node in CG}
9:          Let p = node representing parent(n) in IG
10:         if there is no neighbour x of p such that sim(n,x) > λ then
11:            for all y in IG, if sim(n,y) > μ then add an edge (n,y)
12:            Set id of n as described in IG Construction
13:         end if
14:      else if there is no node x in IG such that sim(n,x) > λ then
15:         for all y in IG, if sim(n,y) > μ then add an edge (n,y)
16:         Set id of n as described in IG Construction
17:      end if
18:   end for
19:   i++
20: end while
```

If the input document set is singleton set then the integrated graph is equivalent to the contextual graph of that document. Addition of a any new document to the set can be reflected in IG by adding its CG as described above and the edge weights are updated accordingly. The integrated graph for the set of documents can also be computed offline and stored. When a query is posed on a document set, its IG can be loaded into memory and $CTrees$ and $SGraphs$ can be constructed as described below.

[2] η is number of id's available for sentences from other documents. We use η = total number of sentences in all documents.

CTree Construction

$CTrees$ rooted at each node in Integrated Graph are constructed as described below. The neighbourhood of a node r is explored and prominent nodes in it are included in $CTree$ rooted at r. This exploration is done in breadth first fashion. Only b(beamwidth) prominent nodes are considered for further exploration at each level. The prominence of a node j is determined by taking the weight of the edge connecting j to it's parent i and it's node score with respect to the query term q into consideration. It is computed as $(\alpha w(e_{ij}) + \beta w_q(j))$. These two factors specify the contribution of the node to the coherence of the summary and the amount of query related information. α is the scaling factor defined in Equation 3. This scaling brings edge weights into the range of node weights and β determines trade-off between coherence and importance of query relevant information.

The exploration from selected prominent nodes (atmost b) is continued until a level which has a node containing a query term ($anchor\ node$) or maximum depth d is reached. All nodes along the path from root to anchor node, along with their siblings are added to the $CTree$. When query term is not found until maximum depth d then $CTree$ for that query term remains empty. If a root node has the query term then root and its adjacent "b" nodes are added to $CTree$ and no further exploration is done.

SGraph Construction

In $CTree$ construction, the direction of the exploration of the graph is determined by the node weights and edge weights. These $CTrees$ formed for each query term rooted at a particular node r are merged to form a $SGraph$. The $SGraph$ at a node, contains all nodes and edges that appear in any one of the $CTrees$ rooted at that node. As $CTrees$ of all query terms are merged to form a $SGraph$, completeness is ensured. As we are merging $CTrees$ rooted at a node, we will have inter connected set of sentences in the summary and hence coherence is preserved.

The $SGraphs$ formed are ranked based on the score computed as per Equation 4. Sentences from the highly ranked $SGraph$ are returned as summary. These sentences are placed in the increasing order of their id's.

Experimental Results

In the experiments, QueSTS was compared with two query specific systems - a baseline and MEAD(Radev, Jing, & Budzikowska 2000). Our baseline system generates summaries by considering only centrality based node weights as per Equation 1 using incremental graph, without generating $CTrees$ and $SGraphs$. Nodes which have high

weights are included in summary. Second system, MEAD[3] is a publicly available feature-based multidocument summarization toolkit. It computes a score for each sentence from a cluster of related documents, as a linear combination of several features. For our experiments, we used centroid score, position and cosine similarity with query as features with 1,1,10 as their weights respectively in MEAD system. Maximal Marginal Relevance (MMR) reranker(Carbonell & Goldstein 1998) provided in the MEAD toolkit was used for redundancy removal with a similarity threshold as 0.6. Equal number of sentences as generated by QueSTS were extracted from the above two systems.

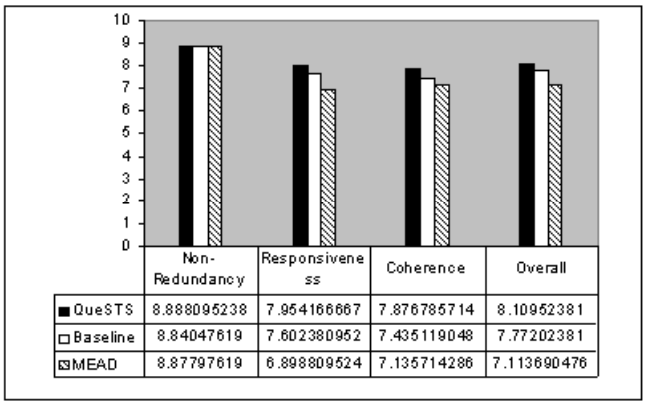
We have experimentally evaluated the summaries generated for 28 queries on wide variety of articles collected from WWW and ProQuest[4], which is a collection of more than 1500 magazines, newspapers, scientific journals, trade magazines, dissertations etc. Top 10 results for a search query from ProQuest and Google were made as a cluster and 11 clusters were taken. Our dataset had a heterogeneous collection of news and technical articles.

We used four criteria to evaluate the generated summaries. They are non-redundancy, responsiveness, coherence and overall performance. Redundancy is defined as unnecessary repetition of facts in summary, responsiveness is measured in terms of the amount of information in the summary that actually helps user to satisfy the information need expressed in the query and coherence determines the information flow in the summary. In our experiments we used $\beta = 1$, $b = 3$, $\lambda = 0.7$ and $\mu = 0.001$ values. Summaries generated by three systems QueSTS, baseline and MEAD were evaluated by a group of 10 volunteers. They were given a set of instructions defining the task and criteria and were asked to rate each summary on a scale of 1(bad) to 10(good) for each criteria without actually seeing the original documents. An average of these ratings for each query was computed and mean of them was calculated. The graph in figure shows that QueSTS performs better when compared to other systems.

MEAD uses MMR principle to handle redundancy, so it does well. As node weights based on centrality are considered to compute query relevance, responsiveness is high for both baseline and QueSTS when compared to MEAD. Supporting sentences were considered in $CTree$ construction, improving QueSTS performance over baseline. As we explore strong contextual relationships in summary construction, coherence is preserved. On the whole QueSTS performed better than others in terms of user satisfaction. Sample summary generated by our system and MEAD is given in Table 1.

[3]www.summarization.com/mead/

[4]http://proquest.umi.com/login



	Non-Redundancy	Responsiveness	Coherence	Overall
QueSTS	8.888095238	7.954166667	7.876785714	8.10952381
Baseline	8.84047619	7.602380952	7.435119048	7.77202381
MEAD	8.87797619	6.898809524	7.135714286	7.113690476

Figure 1: Evaluation Results

Conclusions

In this paper, we presented a novel framework for multi-document summarization system which generates a coherent and intelligible summary. We propose the notion of an integrated graph that represents inherent structure present in a set of related documents by removing redundant sentences. We generate query term specific contextual trees $(CTrees)$ which are merged to form query specific summary graph $(SGraph)$. We have introduced an encoding strategy to order sentences in summary using integrated graph structure. This process of computation has indeed improved quality of summary. We experimentally prove that our approach is feasible and it generates satisfactory summaries.

References

Barzilay, R., and Elhadad, M. 1997. Using lexical chains for text summarization. In *In Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL, Madrid, Spain.*

Bosma, W. 2006. Query-based extracting: how to support the answer? In *Document Understanding Conference*.

Carbonell, J., and Goldstein, J. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 335–336. New York, NY, USA: ACM Press.

Chuang, W. T., and Yang, J. 2000. Extracting sentence segments for text summarization: a machine learning approach. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 152–159. New York, NY, USA: ACM Press.

Edmundson, H. P. 1969. New methods in automatic extracting. *J. ACM* 16(2):264–285.

Erkan, G., and Radev, D. R. 2004. LexPageRank: Prestige in multi-document text summarization. In *EMNLP*.

Fisher, S.; Roark, B.; Yang, J.; and Hersh, B. 2005. OGI/OHSU Baseline Query-directed Multi-

document Summarization System for DUC-2005. *Proceedings of the Document Understanding Conference (DUC)*.

Hovy, E., and Lin, C. 1997. Automated text summarization in summarist. In *In Proceedings of the Workshop on Intelligent Scalable Text Summarization, pages 18–24, Madrid, Spain*.

John M. Conroy, J. D. S., and Stewart, J. G. 2005. CLASSY query-based multi-document summarization. *Proceedings of the Document Understanding Conference (DUC)*.

Luhn, H. P. 1958. The automatic creation of literature abstracts. *IBM Journal of Research Development* 2(2):159–165.

Mihalcea, R., and Tarau, P. 2004. TextRank: Bringing order into texts. In Lin, D., and Wu, D., eds., *Proceedings of EMNLP 2004*, 404–411. Barcelona, Spain: Association for Computational Linguistics.

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

Radev, D. R.; Jing, H.; and Budzikowska, M. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *NAACL-ANLP 2000 Workshop on Automatic summarization*, 21–30. Morristown, NJ, USA: Association for Computational Linguistics.

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

Wan, X.; Yang, J.; and Xiao, J. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, 2903–2908.

Table 1: Query: "Chlorine effects on ozone"

QueSTS Summary: At that height, above 10 kilometers, CFCs and some other gases split apart and loose their destructive chlorine or bromine in the midst of the ozone layer-the shield that protects Earth's surface by absorbing harmful ultraviolet light. That is, chlorine (Cl) reacted with ozone (O_3), generating chlorine monoxide (ClO), which in turn reacted with an oxygen atom to release another chlorine; the net result was that the chlorine was destroying ozone without depleting itself. Yet chlorine was still increasing in the most important place, the stratospheric layer where ozone resides. CFCs nibble away at the layer of ozone that stops most harmful ultraviolet radiation from reaching the earth. But once they reached the mid-stratosphere, above most of the protective layer of ozone, the intense solar radiation broke the CFC molecules apart, releasing chlorine. Foams made with CFC substitutes are less effective as insulators than those made with CFCs.
MEAD (*Query Specific* Summary): ENVIRONMENTAL REGULATION, Remembering the Montreal Protocol: As its 20th anniversary approaches, what can the landmark agreement on controlling CFCs teach those who want to control greenhouse gases? That changed in dramatic fashion with a series of discoveries concerning the global effects of a family of chemicals called chlorofluorocarbons, or CFCs. Ozone Diplomacy: This September will mark the 20th anniversary of the Montreal Protocol on Substances that Deplete the Ozone Layer, an international agreement that set a schedule for freezing and then phasing out the production of CFCs the 1987 treaty, which mandated halving CFC production in industrial countries by 1998, was subsequently revised; CFC production was ended in the United States by 1996. "There is more reason to act on limiting ozone-destroying chemicals now than ever, but there is a sense of sluggishness about doing anything further," says John Passacantando, director of the Washington-based environmental group Ozone Action. The precise make-up of the revitalised ozone layer, especially the vertical distribution of ozone, will depend on three variables: global warming, patterns of air circulation and concentrations of non-CFC gases such as nitrous oxide, which also damages ozone. Further evidence that the chlorofluorocarbons 11 and 12 area greater threat to the ozone layer in the stratosphere than any other chemical compound is citeyeard in a report published by the Department of the Environment's Central Unit on Environmental Pollution CUEP.