A NOVEL APPROACH TO INFORMATION EXTRACTION AND SENTENCE ORDERING IN MULTI DOCUMENT SUMMARIZATION

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree

of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

AMIT GUSAIN

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2008

[illegible]

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in this dissertation report, entitled "**A Novel Approach to Information Extraction and Sentence Ordering in Multi Document Summarization**", is being submitted in partial fulfillment of the requirements for the award of the degree of **Master of Technology** in **Computer Science & Engineering**, in the Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee and is an authentic record of my own work, carried out from July 2007 to June 2008, under guidance and supervision of **Dr. R. C. Joshi**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee.

The results embodied in this dissertation have not been submitted for the award of any other Degr or Diploma.

Date:

Place: Roorkee

[illegible]

(AMIT GUSAIN)

CERTIFICATE

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

[illegible]

(Dr. R. C. JOSHI)
Professor, E&C Department,
Indian Institute of technology,
Roorkee – 247 667, (INDIA)

ACKNOWLEDGEMENT

At the outset, I express my heartfelt gratitude to **Dr. R. C. Joshi**, Professor, Department of Electronics and Computer Engineering at Indian Institute of Technology Roorkee, for his valuable guidance, support, encouragement and immense help. I consider myself extremely fortunate for getting the opportunity to learn and work under his able supervision. I have deep sense of admiration for his innate goodness and inexhaustible enthusiasm. It helped me to work in right direction to attain desired objectives. Working under his guidance will always remain a cherished experience in my memory and I will adore it throughout my life.

I also extend my sincere thanks to **Dr. D. K. Mehra**, Professor and Head of the Department of Electronics and Computer Engineering, and **Mr. Raj**, Lab Assistant, Information security Lab, Department of Electronics and Computer Engineering, for providing facilities for the work.

My sincere thanks are also due to rest of the faculty in the Department of Electronics and Computer Engineering at Indian Institute of Technology Roorkee, for the technical know how and analytical abilities they have imbibed in us which have helped me in dealing with the problems I encountered during the project.

I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample morale support and valuable suggestions. Finally, I would like to extend my gratitude to all those persons who directly or indirectly helped me in the process and contributed towards this work.

I dedicate this work to my family for their support and encouragement throughout my life.

[illegible]

Amit Gusain
M. Tech. (CSE)

ABSTRACT

With the rapid growth of the World Wide Web and electronic information services, the amount of information is growing at an incredible rate. One problem that arises due to this exponential growth is the problem of information overload. No one has time to read everything, yet we often have to make critical decisions based on what we are able to assimilate. With summaries, we can make effective decisions in less time. Thus the technology of automatic text summarization is becoming essential to deal with the problem of information overload.

Text summarization is the process of extracting the most important information from a single document or from a set of documents to produce a short and information rich summary for a particular user or task. Multi-document summarization is an automatic procedure for extraction of information from multiple texts written about the same topic. Most of the MDS systems have been based on an extraction method, which identifies key textual segments (eg sentences or paragraphs) in source documents and selects them for the summary. It is important for such MDS systems to determine a coherent arrangement (ordering) of the textual segments extracted from the source documents in order to reconstruct the text structure for summarization.

In this dissertation work we have focused on the two key tasks of the summarization, information extraction and sentence ordering. A multi document summarization method based on frequency of bi-grams (window of size 2 words) is used for the information extraction task. As the sentences are selected based on their importance from the documents they lose the cohesion and the ordering of the information in the summary thus loosing the readability of the summary. To deal with this problem, we propose a new method for sentence ordering based on the types of the sentences. Our results show that the proposed multi document summarizer approach works significantly well in extracting important content units and improving the readability of the summary.

CONTENTS

List of Figures

List of Tables

Chapter 1
INTRODUCTION

1.1 Introduction

In the recent years there has been a well-publicized explosion of information available over the Internet, and a corresponding increase in its usage. Everyday we are bombarded with reams of information in all forms (eg. e-mails, papers, books, magazine articles, web pages etc.). The storage cost is very low and the storage capacity is almost limitless, resulting in billions of documents available on web. A study conducted by the University of California at Berkeley estimates that almost 800 Megabytes of stored information are produced per person, per year [1]. The production of information has increased to the extent that we are now seen to be in the midst of an information explosion. As a result of this explosion of information, we are experiencing a state called "Information Overload". The growing number of electronic articles, magazines and books that are available on-line today are putting more pressure on people, as they struggle with information overload. No one has time to read everything, yet we often have to make critical decisions based on what we are able to assimilate. The technology of automatic text summarization is becoming indispensable to deal with this problem and reduces the pressure of reading full articles. Summarization, which is the art of abstracting key content from one or more information sources, has become an integral part of everyday life. For example people keep them up to date of world affairs by listening to news (summary of world affairs). They base investment decisions on stock market updates (summary of the market). They go to movies largely on the basis of reviews (summary of comments).

Automatic summarization [2] is the process of distilling the most important information from a source to produce an abridged version for a particular user or task. Human generates a summary of a text by understanding it with the deep semantic processing using huge domain/background knowledge. It is too difficult for the current computer to simulate this process. Therefore, most automatic summarization programs analyze a text statistically and linguistically, determine important sentences, and generate a summary text from these important sentences. The goal of the automatic text summarization is to provide a user with a presentation of the substance of a body of material in a coherent and concise form to save time and effort. Ideally, a summary should contain only the "right" amount of the interesting information and it should omit all the redundant and "uninteresting" material. The summary produced by automatic summarization can be of two types - generic or user specific [3]. The generic summaries contain the overall most salient information from the original documents while the user specific summaries contain the most relevant information depending upon the choice and interests of the user.

Automatic text summarization can be broadly categorized in two types based on the number of source documents: Single Document Summarization and Multi Document Summarization (MDS) [4]. As the name suggests in single document summarization there is only one large source document, while in case of multi document summarization the information is distributed over multiple source documents. Single document summarization is easy as compared to multi document summarization task. As in single document summarization there is no issue of multiple languages, multiple input format, writing style, redundancy of information etc [5]

The multi-document summarization [6] task has turned out to be much more complex than summarizing a single document, even a very large one. This difficulty arises from inevitable thematic diversity within a large set of documents. These documents can be in different languages, written by different authors having different background knowledge and different document formats. A good summarization technology aims to combine the main themes with completeness, readability, and conciseness [7]. An ideal multi-document summarization system does not simply shorten the source texts but presents information organized around the key aspects to represent a wider diversity of views on the topic. When such quality is achieved, an automatic multi-document summary is perceived more like an overview of a given topic.

There are two types of situations in which multi-document summarization would be useful: (1) the user is faced with a collection of dissimilar documents and wishes to assess the information landscape contained in the collection, or (2) there is a collection of topically-related documents, extracted from a larger more diverse collection as the result of a query, or a topically-cohesive cluster. In the first case, if the collection is large enough, it only makes sense to first cluster and categorize the documents, and then summarize each cohesive cluster. Hence, a "summary" would constitute of a visualization of the information landscape, where features could be clusters or summaries thereof. In the second case, it is possible to build a synthetic textual summary containing the main point(s) of the topic, augmented with non-redundant background information and/or query-relevant elaborations. This is the focus of dissertation work reported here, including the necessity to represent the information in readable ordering the selected from the multiple related documents.

Multi-document summarization creates information reports that are both concise and comprehensive. With different opinions being put together & outlined, every topic is described from multiple perspectives within a single document. While the goal of a brief summary is to simplify information search and cut the time by pointing to the most relevant source documents [8], comprehensive multi-document summary should itself contain the required information, hence limiting the need for accessing original files to cases when refinement is required. Automatic summaries present information extracted from multiple sources algorithmically, without any editorial touch or subjective human intervention, thus making it completely unbiased.

The multi document summarization can be categorized along two different dimensions: abstract-based [3, 9, 10] and extract-based [11, 12, 13, 14]. An extract-summary consists of sentences extracted from the document while an abstract-summary may employ words and phrases that do not appear in the original document. The extractive summarization tries to select a number of indicative sentences, passages or paragraphs from the original document according to a target summarization ratio, and then sequence them together to form a summary. The abstractive summarization, on the other hand, tries to produce a concise abstract of desired length that can reflect the key concepts of the document. The latter seems to be more difficult, and most of the recent approaches have focused more on the extraction based summarization. In this dissertation we have focused on the information extraction based generic multi document summarization. Information extraction is a shallow approach in which, statistical heuristics are used to identify the most salient sentences of a text. Information extraction [14] is a low-cost approach compared to more knowledge-intensive deeper approaches which require additional knowledge bases such as linguistic knowledge. In short, information extraction works as a filter which allows only important sentences to pass. Sentence abstraction is the natural language processing task of automatically generating natural language sentences from a set of source documents [16]. The sentences generated by this technique contain the over-all information of all the source documents. But it is very difficult to achieve a high degree of accuracy in sentence abstraction tasks as the computer systems do not have deep semantic knowledge and ability to understand like humans have.

Most MDS systems are based on the information extraction method, which extracts the most important sentences in source documents and include them into the summary document [15]. So for such system it is very much important to provide a coherent ordering of the sentences extracted from different source documents in order to make the summary meaningful. Sentence ordering, which affects coherence and readability, is of particular concern for multi-document summarization, where different source articles contribute sentences to a summary.

In this dissertation work techniques for the above two key tasks (Information Extraction and Sentence Ordering) of the generic multi document summarization has been proposed. Information extraction based on bi-gram frequency (discussed in chapter 3) has been selected as a method to extract the important content out of the multiple source documents. A sentence ordering technique which is based on the sentence types (discussed in chapter 4) and the chronological information of the documents has also been proposed in this dissertation.

1.2 Statement of the Problem

The dissertation work can be divided in to solving two problems of the multi document summarization processes: first, identifying the most important information that ought to be included in the summary, and second, ordering the information that has been identified in the first process to increase the readability of the summary. The first process is referred to as the Information extraction process and the second is referred as the sentence ordering process.

1.3 Organization of the Dissertation

This report comprises of six chapters including this chapter that gives the overview of the automatic summarization systems and discuss about the need motivation for such systems. It also summarizes the problem statement for the dissertation work. Rest of the dissertation report is organized as follows.

Chapter 2 gives the background and literature survey of the various MDS systems in the field of information extraction. It also presents the existing methods for sentence ordering. It also describes the framework of generic multi document summarization system and the research gaps in information extraction and sentence ordering techniques.

Chapter 3 gives the description of proposed framework for the information extraction module. It also describes the tools used for preprocessing of the input text and creating and reducing the term-term matrix.

Chapter 4 describes the proposed framework for the sentence ordering module for the MDS system with detailed description of the sentence types and dictionaries used.

Chapter 5 discusses the evaluation metrics, the data set used for the testing purpose, the performance of the system, the tables and graphs depicting the performance. It also presents summaries produced by our system along with the human generated summaries for the same set of documents.

Chapter 6 concludes the findings of the dissertation work and gives suggestion for future work.

Chapter 2
Background and Literature Survey

2.1 A Generic Framework for Multi Document Summarization (MDS)

The process of multi document summarization is not a single shot process rather it is divided in to a series of sub processes, which includes extraction of information, information representation, sentence ordering, and summary generation. The main problems an MDS need to solve are information extraction (i.e. deciding and extracting from the input documents that are important enough to be included in a summary) and sentence ordering (i.e., deciding the order in which the extracted sentences from the input documents should appear in the summary). Systems that go beyond sentence extraction and use generation techniques to reformulate or simplify the text of the original articles also need to decide which simplified sentences should be chosen. So information extraction is an essential component for all multi document summarizers. When sentences are taken out of context and placed one after another in automatic summaries, they may convey meaning that are not at all intended in the original text, presenting misleading or false information. So sentence ordering is also the important component of the MDS systems.

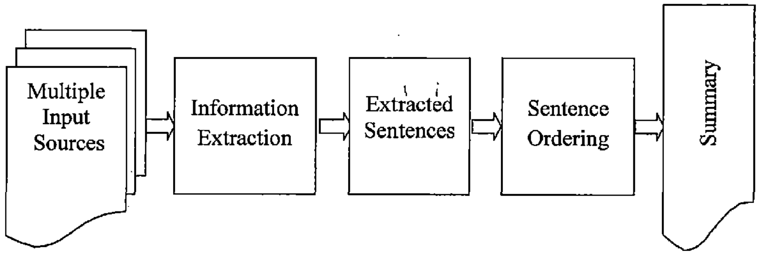

Figure 2.1: Framework for Multi Document Summarization System

The generic framework for the Multi Document summarization system is shown in Figure 2.1. It consists of two processes: Information extraction module and Sentence ordering module. Information Extraction process, takes a cluster of documents with the same theme as input and extracts the important sentences (or information) to be included in the summary. Sentence ordering process coherently orders the sentences which the information extraction process selects.

2.2 Information Extraction Approaches

There are several ways in which one can characterize different approaches to information extraction based summarization. One useful way is to examine the level of processing. Based on this, summarization can be characterized as approaching the problem at the surface, entity, or discourse levels [2].

Surface-level approaches [11, 12, 13, 17, 18] tend to represent information in terms of shallow features which are then selectively combined together to yield a salience function used to extract information. These features include frequency, location, background, cue words and phrases. Entity-level approaches [20, 21] build an internal representation for text, modeling text entities and their relationships. These approaches tend to represent connectivity in the text to help determine what is salient. Relationships between entities include similarity, proximity, co-occurrence, thesaural relationships among words (synonymy, antonymy, parts-of relations), logical relations (agreement, contradiction, and consistency) syntactic relations. Discourse-level approaches [16, 22] model the global structure of the text, and its relation to communicative goals. This structure can include format of the document, threads of topics as they are revealed in the text, and rhetorical structure of the text, such as argumentation or narrative structure. These are the primary examples of the approaches, and many systems adopt a hybrid approach (e.g., taking a discourse level approach where the smallest segments are surface strings or entities). Table 2.1 shows the list of features and their description for all three approaches in brief [2].

Approach	Feature	Description
Surface Level	Frequency	Statistically salient terms
	Location	Sentence position in paragraph, paragraph position in text
	Title	Headline, topic
	Query	User interest
	Cue words/ phrases	Bonus terms, stigma terms
Entity level	Similarity	Vocabulary overlap
	Proximity	Distance between text units
	Thesaural relationship	Synonymy, parts-of relationship
	Syntactic relationship	Based on parse trees
Discourse level	Format of document	Topic classification
	Threads of topics	Topic segmentation
	Document discourse structure	Rhetorical structure

Table 2.1: Surface, entity and discourse-level features.

2.3 Desired Features of a MDS System

The task of a multi document summarization system is to generate a short paragraph that presents the important information present in the given clusters of input documents on same topic in a coherent (properly ordered) form. Besides keeping the summary short, the summarization process must also preserve the information contained in the original document [8]. The most desired features of a multi document summarization system are *text compression, information preservation, text cohesion and redundancy removal.*

Text Compression: The main feature that a multi-document summarization system is required to do is shortening of the original text as per the interest of the user. This would naturally mean the system should ideally be able to adjust the length of the output according to the compression rate given by the user. For example, if the compression rate given is 20%, then the system must compress a 1000 sentences document to 200 sentences document.

Information Preservation: While keeping the output text short, the system should be able to preserve as much salient information as possible. Hence the system needs to make decisions to choose from the document cluster, competing sentences to be included in the summary. The sentences with better salience scores than a given threshold value (lower threshold value for information extraction phase) are selected for summary. The threshold value depends on the size and structure of the source documents.

Summary Cohesion: An ideal multi-document summarization system must produce a syntactically correct and coherent summary. There should not be abrupt shifts of topics in the summary and the flow and order of the summary sentences must be cohesive. This requires the summary should not contain grammatically incorrect sentences and must be ordered in a readable form. Unless the system is pure sentence picker, it should take care of the syntax of the sentences included in the resulting summary.

Redundancy Removal: The multi-document summarization system should have the ability to eliminate the redundancy among the sentences in the summary. There should not be more than one sentence depicting the same information. So a sentence, that is to be included in the summary, must be first cross checked with the sentences already included in the summary for any kind of redundancy.

2.4 Types of Multi Document Summarization Systems

Different MDS systems use different measures in assigning the salience score to the sentences. Based on the methods the MDS systems employ in assigning salience score to the sentences, they can broadly be classified into three categories as centroid based, clustering based and graph based summarization. Here we briefly describe the general methods employed in assigning salience scores for the sentences in each of these three categories.

2.4.1 Clustering Based MDS

One of the first and very popular approaches to MDS was cluster topically related sentences from the input and select one sentence from the cluster as a representative of the topic in the summary [31]. These summarizers obviously try to exploit frequency on the sentence level, clusters with more sentences considered more important. Again, a hidden parameter can change the results considerably since if lower similarity between sentences in the cluster is required, bigger clusters can be formed, but the sentences in them will not be tightly related on the same topic. Such an approach assigning importance to sentences also deals directly with the problem of duplication removal: since only one sentence per cluster is chosen, the summary would not include repetition. Interestingly the size of the cluster (equivalent to sentence frequency), did not lead to good information extraction performance. The problem was addressed by adding in the weighting of term frequency (tf) and inverse document frequency (idf). The addition of such information, which incorporates in the cluster score, the frequency also of the words in the sentences, leads to much better results in information extraction.

2.4.2 Centroid Based MDS

Radev et al. [17] described an extractive multi document summarizer (MEAD) which chooses a subset of sentences from the original documents based on the centroid of the input documents. For each sentence in a cluster of related documents, MEAD computes three features and uses a linear combination of the three to determine the most important sentences. The three features used are centroid score, position, and overlap with first sentence (or the title).

The centroid score C_i is a measure of the centrality of a sentence to the overall topic of a cluster. The position score P_i which decreases linearly as sentence gets farther from the beginning of the document, and the overlap with first sentence score F_i which is the inner product of the tf-idf weighted vector representations of a given sentence and the first sentence (or title) of the document. All three features are normalized (0-1) and the overall score for a sentence Si is calculated as

$$W(Si) = W_c * C_i + W_p * P_i + W_f * F_i,$$

Where W_c, W_p, and W_f are the individual weightage given to each type of features respectively. MEAD discards sentences that are too similar to other sentences. Any sentence that is not discarded due to high similarity and which gets a high score is included in the summary

2.4.3 Graph Based MDS

Some of the most newly developed summarizers are those that reduce the problem of summarization to graph problems, notably using the Page-Rank algorithm. Of these, the most successful application to multi document summarization was that of Erkan and Radev [22]. In their LexRank algorithm, each sentence defines a node in the text graph. To define edges in the graph, the cosine similarity between two sentences is computed and an edge is added between the nodes representing the two sentences if the similarity exceeds a predetermined threshold. Thus the edges are defined for sentences that share the same words. The Page-Rank algorithm is then used iteratively to compute the rank of each sentence as a function of the number of neighbors and the importance of the neighbors of each node. The iterations distribute the weight across the graph, and quickly explain that the iterative spreading of importance in the graph is similar to voting process: sentences from the entire graph vote for the sentences with which they share word overlap. Of course, such a voting procedure can be achieved by a direct frequency count, rather than distributing information little by little through the nodes. So the Page-Rank algorithm can be seen as a complex (unobservable) function that assigns weights to sentences based on the frequency of words that appear in the text. In order to avoid repetition, sentences that are assigned high importance, but are similar to more important sentences are not included in the summary.

All the above systems select sentences based on importance derived from some complex formulas which directly or indirectly uses the frequency of the words for the computation of importance. But no system has studied the contribution of the frequency of the bi-gram units in the multi document summarization.

2.5 Related work in Information Extraction

There have been a number of researches and development budgets [2] devoted to automatic text summarization. The United States (e.g., DARPA), the European Community and Pacific Rim countries have identified text summarization as a critical research area, and are investing in it. Text summarization is also increasingly being exploit in the commercial sector, in telecommunication industry (e.g., BT's ProSum), in filters for web based information retrieval (e.g. Inxight's summarizer used in AltaVista Discovery), and in word processing tools (e.g.., Microsoft's AutoSumarize). In addition to the traditional focus of automatic abstracting (of scientific and technical text) to support information retrieval, researchers are investigating the application of this technology to a variety of new and challenging problems, including multilingual summarization, multimedia news broadcasts, and providing physicians with summaries of on-line medical literature related to patient's medical record. As the information overload problem has grown, and people become increasingly mobile and information-hungry, new applications for text summarization can be expected.

The early systems for text summarization were developed in late 1950's [11] characterized by surface-level approaches. First entity-level approach based on syntactic analysis and the use of the location feature was introduced later by Edmundson [12]. In 1970s, there was renewed interest in the field with extensions being developed to the surface-level approach to include the use of cue phrases (bonus versus stigma items). The late 1970s saw the emergence of more extensive entity-level approaches as well as the first discourse-based approaches on story grammars. The 1980s enjoyed an explosion of a variety of different work, especially entity-level approaches based on artificial intelligence such as the use of scripts, logic and production rules, semantic networks, as well as hybrid approaches. The period of late 1990s represents a renaissance of the field, with three types of approaches being explored very aggressively, heightened by government and commercial interest. The work done during this period [3, 13, 14, 16] has almost exclusively focused on extracts rather than abstracts, along with a renewed interest in earlier surface-level approaches.

Summarization Feature / **Summarizer System**	Term frequency	Location	Sentence length	Cue phrases	doc format	Generic focus	Query focus	Controllable compression	multilingual	trainable	>= 5 file format	Evaluated
AutoSummarizer (MS Word' 97)	*Y*					*Y*						
Context	*Y*					*Y*		*Y*				*Y*
Data Hammer	*Y*							*Y*	*Y*	*Y*		
DimSum	*Y*							*Y*	*Y*	*Y*		*Y*
Extractor	*Y*					*Y*			*Y*	*Y*		*Y*
GESummarizer						*Y*	*Y*	*Y*				*Y*
Intelligent Miner	*Y*	*Y*				*Y*		*Y*				*Y*
IntelliScope	*Y*	*Y*			*Y*	*Y*	*Y*	*Y*	*Y*		*Y*	
InText							*Y*	*Y*				
InXight Summarizer plus	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*		*Y*	*Y*	*Y*	*Y*	
ProSum	*Y*	*Y*	*Y*									*Y*
Search 97 Developer's Kit	*Y*	*Y*	*Y*	*Y*		*Y*	*Y*	*Y*				
SMART								*Y*				*Y*
SUMMARIST	*Y*					*Y*	*Y*		*Y*			
TextNet 32	*Y*					*Y*	*Y*	*Y*				
TextAnalyst 2.0							*Y*					

Table 2.2: Features Comparison of Commercial Summarizer Systems

In the 2000s the focus of the interest of the researches shifted towards Multi document summarization. A variety of multi-document summarization methods have been developed recently. The centroid-based method [17] is one of the most popular extractive summarization methods. MEAD is an implementation of the centroid-based method [19]. NeATS [23] uses sentence position, term frequency, topic signature and term clustering to select important content, and use MMR [13] to remove redundancy. Recently graph based have been proposed to rank sentences or passages. LexPageRank [22] and Mihalcea and Tarau [21] are three such systems using algorithms similar to Page-Rank to compute sentence importance. Table 2.2 shows the comparison among the different commercial summarizers based on various summary features [2]. Where *Y* represents that the system implements the feature

2.6 Significance of Sentence Ordering

Most often, the extractive summaries produced from multiple source documents suffer from an array of problems with respect to text coherence and readability, like dangling references, irrelevant context cue information, etc. Many approaches [25, 26, 27, 28, 29] have been proposed to deal with problems, including co-reference resolution, temporal information recovery and removal of contextual phrases by sentence compression. But after these post processing steps, even if each individual sentence might be interpretable in isolation, it still does not mean that sentences gathered from different sources as a whole will be easy to understand. Interdependence between sentences greatly affects reader's understanding. Therefore, it is important to consider sentence ordering of extracted sentences in order to reconstruct discourse structure in a summary. Sentence ordering, which determines the sequence in which to represent a set of pre-selected sentences, is a critical task both for text summarization and natural language generation. The problem of how to structure the selected information to form a fluent summary has received very little attention until recently. In single document summarization, summary sentences are typically arranged in the same order as they were in the original full document, although it was found that human summarizers do sometimes change the original order [24]. In multi-document summarization, sentences are selected from multiple documents and no complete ordering from a single document is available, so most common approaches involve ordering by the original article publishing time or ordering sentences based on their content importance score from the extraction stage

[30]. Several approaches have been taken in solving the information ordering task in multiple document summarization, all of which follow the assumption that the summary structure also follows the structure of the original document set, since multi-document summary captures the main contents among the document clusters.

2.7 Sentence Ordering Techniques

The first systematic research on sentence ordering was done by Barzilay, et al [24]. They provided a corpus based methodology to study ordering and conducted experiments which show that sentence ordering significantly affects the reader's comprehension. They also evaluated two ordering strategies: Majority Ordering which orders sentences by their most frequent orders across input documents and Chronological Ordering which orders sentences by their original article's publishing time. They then introduced an augmented chronological ordering with topical relatedness information that achieves the best results. The augmented strategy used majority and chronological constraints to define the pair wise relations between sentences. Barzilay then identified the final order of sentences by finding a maximal weighted path in a precedence graph [24]. Table 2.3 [31] shows four sentence ordering techniques with the features and scoring method they used. Table 2.4 [31], gives brief description about the data and evaluation method of each of the four sentence ordering techniques. An unsupervised probabilistic model has been suggested by Lapata [25] for text structuring that learns ordering constraints from sentences represented by a set of lexical and structural features. It assumes the probability of any given sentence is determined by its previous sentence and learns the transition probability from one sentence to the next from the BLLIP corpus based on the Cartesian product between two sentences defined using the following features: verbs and their precedent relationships; nouns (entity-based coherence by keeping track of the nouns); and dependencies (structure of sentences). The overall ordering of the sentences in the summary is learned by greedily searching for a maximal weighted path through the graph. Based on the experimental results, she finds that entity-based coherence and the verb-noun structure features are significantly better than any other features.

	Brazilay 2002 [24]	Lapata 2003 [25]	Brazilay & Lee 2004 [29]	Okazaki, et al 2004, 2006 [26, 27]
Hypothesis	1. Sentence order do impact the user comprehension 2. Multiple acceptable ordering for one document 3. Topical related sentences share adjacency relation.	Local coherence can be captured through the probability of lexical and syntactic features of sentence based on the previous sentence, learn text structure for a specific domain	Word distributional patterns characterize various types of discourse (content structure) which can be captured using HMM	Use the machine learning framework to incorporate the four ordering criteria to capture the contingency between two sentences
Rank/ Search	Search through weighted precedence graph	Simple greedy search through weighted graph	Ranking by HMM	Agglomerative hierarchical clustering with the ordering information retained
Features	Majority ordering, chronological ordering, topical relatedness augmented chronological ordering	Verbs, nouns, structure dependencies	State: topic clustering, Transitional Pr: sentence position in the original article	Chronological sequence, topical relatedness, precedence and succession

Table 2.3: Sentence Ordering Techniques

Barzilay and Lee [29] have proposed domain-specific content models to represent topics and topic transitions for sentence ordering. They learn the content structure directly from un-annotated texts via analysis of word distribution patterns based on the idea that "various types of [word] recurrence patterns seem to characterize various types of discourse". The content models are Hidden Markov Models (HMMs) wherein states correspond to types of information characteristic to the domain of interest, and state transitions capture possible information-presentation orderings within that domain.

		Brazilay 2002	Lapata 2003	Brazilay & Lee 2004	Okazaki, et al 2004, 2006
Data	Corpus	25 sets of topics, each has 2-3 news articles reporting the same event	BLLIP corpus (30 M words) + Brazilay 2002 corpus	5 domains (earth-quake, finance, etc) Each domain has 100 training/ 100 testing/ 20 development set	TSC-3 corpus (Japanese), containing 30 sets of human ordered extracts for multiple document summarization relevant to questions
	Input	Manually selected sentences as extract	Human written articles	Human written articles	Automated extracted sentences for summary
	Length	8.8 sentences	15.3 sentences	12 sentences	15 sentences
Evaluation	Human	3 level grading: poor, fair, good	Human produce summary for upper bound of Kendall's tau	No	4-scales: perfect, acceptable, poor, unacceptable
	Automatic	No	Kendall's tau (Distance between model and original article)	OSO prediction rate, pair-wise comparison	Spearmen's and Kendall's Tau correlation + continuity metrics

Table 2.4: Data and Evaluation for Sentence Ordering Techniques

Bollegala, Okazaki and Ishizuka [26] provide a novel supervised learning framework to integrate different criteria. They also propose two new criteria precedence and succession developed from their previous work [27]. A fundamental assumption for the precedence criteria is that each sentence in newspaper articles is written on the basis that pre-suppositional information should be transferred to the reader before the sentence is interpreted. The opposite assumption holds for the succession criteria. They define a precedence function between two segments (a sequence of ordered sentences) on different criteria and formulate the criteria integration task as a binary classification problem and employ a Support Vector Machine (SVM) as the classifier. After the relations between two textual segments are learned, they then repeatedly concatenate them into one segment until the overall segment with all sentences is arranged.

Barzilay and Lapata [28] introduce an entity-based representation of discourse and treat coherence assessment as a ranking problem based on different discourse representations. A discourse entity is a class of co referent noun phrases. They use a grid to represent a set of entity transition sequences that reflect distributional, syntactic, and referential information about discourse entities. A fundamental assumption for this method is that the coherence on the level of local entity transitions is essential for generating globally coherent texts. They then take as input a set of alternative renderings of the same article and rank them based on the local coherence. The ranking problem is solved using the search techniques on a Support Vector Machine constraint optimization problem.

2.8 Research Gaps

2.8.1 Information Extraction

Most of the extraction based multi-document summarization systems take advantage of the frequency of individual words. The more number of times a word occur in the source documents increase the chances of it to be included in the summary. The term frequency [33] is the prime feature in summarization for the tf-idf based multi-document summarization systems. Here TF represents the term frequency that is the frequency of a word in a document, and IDF represents the inverted document frequency that is the distribution of a term in the whole corpus of data and is equal to the number of documents which contains the term divided by total number of documents in the corpus. The content that appears frequently in the input has a higher likelihood of being selected a human summarizer for inclusion in a summary. It is observed that high frequency words from input are very likely to appear in the human summary. This confirms that unigram (individual word) frequency is one of most important the feature that impact a human's decision to include specific content in a summary. But the co-occurrence of the individual words in the inputs and the human summaries does not necessarily entail that the same facts have been covered. A better granularity for such investigation is the sequence of such individual words, such as the summary sentences. Thus the overlapping of a sequence of words (or a sentence) from inputs with the human generated summary confirms that both the documents contain same information. Almost all of the systems have used the unigram frequency for assigning salience scores none has selected the frequency of more than single words which conveys more meaning for the assignment of salience score.

2.8.2 Sentence Ordering

When producing a summary, any multi-document summarization system has to choose in which order to present the output sentences. The first algorithm, Majority Ordering (MO), relies only on the original orders of sentences in the input documents. This algorithm can be used to order sentences accurately if we are certain that the input texts follow similar organizations. This assumption may hold in limited domains where documents have a fixed organization of the information. Looking at the daily statistics of scientific texts, we notice that there are several clusters which contain more than 20 and up to 70 articles to be summarized into single summaries. With such a big number of input articles, we cannot assume that they will all have similar ordering of the information. MO's performance critically depends on the agreement of orderings in the input texts, hence it can not fit all types of input data. The second one, Chronological Ordering (CO), uses time-related features to order sentences and places the sentence in the temporal order of their occurrence. Assigning a date to a reaction, or more generally to any sentence conveying background information, and placing it into the chronological stream of the main events does not produce a logical ordering. The ordering of these themes is, therefore, not covered by the CO algorithm. Furthermore, some sentences cannot be assigned any time stamp. For instance, the sentence, "The vast, sparsely inhabited Xinjiang region, largely desert, has many Chinese military and nuclear installations and civilian mining." describes a state rather than an event and, therefore, trying to describe it in temporal terms is invalid. Thus the ordering cannot be improved at the temporal level. Another drawback with chronological ordering is that summaries generated by this algorithm contains abrupt switches of topics and are generally incoherent. With these shortcomings of the traditional ordering algorithms in mind we propose a new ordering algorithm based on the various types of sentences found in scientific texts, these sentence types are described in detail in chapter 4.

Chapter 3
Proposed Design for Information Extraction Module in MDS

3.1 Framework for Information Extraction Module

Information Extraction module is the first phase of the proposed multi document summarization system. Information extraction (IE) is a type of information retrieval whose goal is to automatically extract structured information, i.e. categorized and contextually and semantically well-defined sentences from a certain domain, from unstructured machine-readable documents. Information extraction is an important component for all the multi document summarizers. In past different systems for MDS used different techniques to calculate the importance of the sentences in the document but all these techniques are based on the features that are related to the individual word tokens. So in past all the systems have focused on very low level granularities (individual words) as discussed in previous chapter for the information extraction module. In this dissertation high level of granularities (sequence of important words) for the information extraction module have been used. We consider the pair of important words (feature terms) as a bi-gram unit.

1. *More than 100 people were killed when a Sudanese Airbus burst into fire after airport in poor weather on Tuesday.*
2. *Around 100 people were killed when a Sudanese Airbus burst into fire after landing at Khartoum airport*
3. *A Sudanese Airbus burst into flames after landing in Khartoum airport overnight in bad weather, killing 100.*
4. *The Sudanese Airbus carrying 214 people veered off the Khartoum airport in a thunderstorm and burst into flames late Tuesday.*

Figure 3.1: Headlines from different newspapers for same event

The sequence of content words conveys more precise meaning than a single content word. Figure 3.1 shows the importance of the bi-gram units over the unigram words. It includes sentences taken from various new papers headlines for the same event. The actual summary (headline) for the above set of sentences is *100 people killed in Sudanese Airbus crash.* But this is assigned by human experts and is not obtained automatically. Our automatically generated bi-gram units for this are *100 people, people killed, Sudanese Airbus, Airbus burst, Khartoum airport* etc, Each bi-gram unit convey more precise meaning than each content word. For example *Khartoum Airport* together gives a meaning that we are talking about the Khartoum Airport. While the individual words like airbus and burst do not automatically suggest the same, it can be deduced from individual words that we are talking about either Khartoum city or Airports or both. This makes the bi-gram units better choice for the information extraction task. So if bi-gram units are chosen for assigning the salience scores to the sentences the MDS system will produce better summaries.

The proposed framework for the information extraction module is shown in Figure 3.2. The framework consists of three basic stages: Input, Processing and Scoring

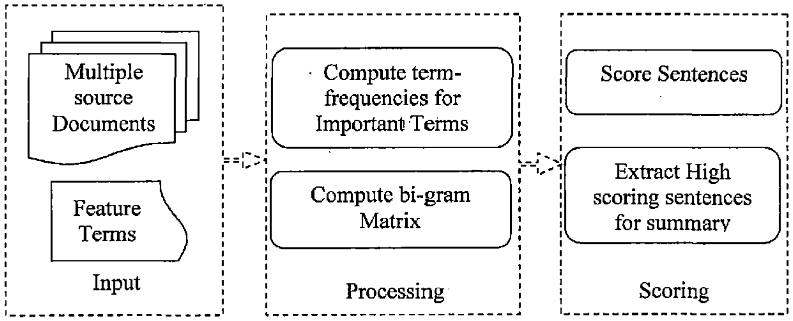
Multiple source Documents
Feature Terms
Input
Compute term-frequencies for Important Terms
Compute bi-gram Matrix
Processing
Score Sentences
Extract High scoring sentences for summary
Scoring

Figure 3.2: Proposed Framework for Information Extraction

The first stage inputs the set of source documents that are clustered around the same topic or subject or event and a list that consist the most common and important words related to the topic or the event known as Feature Terms. The next stage computes the frequency of bi-gram unit (window of size two feature terms). The final stage ranks the sentences based on the terms they contained and their term-term score, and finally the high ranking sentences are selected for inclusion in the summary.

3.2 Implementation of Information Extraction Module

The algorithm for the information extraction module is shown in stepwise fashion in Figure 3.3. The next subsections discuss about each block of the proposed framework.

```
Begin:
    Step 1: Input the set of source documents and the feature term list
    Step 2: Compute the term frequencies for the feature terms and other
            important words (candidates for feature terms) in the input documents
    Step 3: Prepare a term-document matrix consisting the frequency value of
            individual term against each document
    Step 4: Compute the frequency score for each bi-gram (pair of important
            words)
    Step 5: Assign the salience score to the sentences based on the bi-gram
            frequency score for the bi-gram units present in the sentences.
    Step 6: Select the sentences with high scores for the inclusion in summary.
End
```

Fig 3.3: The Algorithm for the Information Extraction Module

3.2.1 Input and Preprocessing

The first stage inputs the set of documents that are clustered around the same topic or subject or event and a list that consist the most common and important words related to the topic or the event known as list feature terms. The feature terms can be given by the authors of the documents or it can be generated on the basis of the background knowledge in that particular topic, the feature terms list can be blank too in this stage as more feature terms are computed in the later stages by selecting the frequent terms in the documents on that topic.

3.2.2 Computing Bi-Gram Matrix

The next stage (Figure 3.4) computes the frequency of co-occurring of each pair of terms which in our case forms the bi-gram units. This stage is divided into the following subtasks.

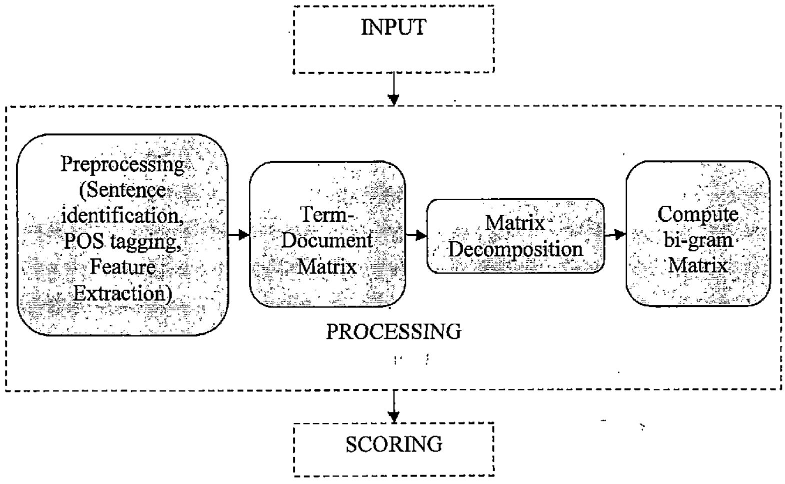

Figure 3.4: Computing bi-gram Matrix

Preprocessing: The documents obtained from the input stage are first preprocessed to convert them to simple machine readable format (plain text format). This is required to identify each sentence unit easily for the extraction and scoring tasks. The preprocessing also includes the POS tagging of the documents and the extraction of feature terms.

POS Tagging: A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc. Parts of speech tags are assigned to the text using Stanford POS tagger, an open source natural language processing library tool [45]. This software is a Java implementation of the log-linear part-of-speech (POS) taggers described in [34] and demonstrates following ideas: (i) explicit use of both preceding and following tag context via a dependency network representation, (ii) broad use of lexical features, including jointly conditioning on multiple consecutive words, (iii) effective use of priors in conditional log linear models and (iv) fine grained modeling of unknown word features. Using these ideas together the resulting tagger gives 97.24 % accuracy on the Penn Treebank corpus. Table 3.1 shows the Penn Treebank Tagset with description and example of each tag. The POS tagged text is then used for the theme (Feature term) extraction. We consider only (Nouns, Verbs, and Adjectives) as the feature terms for their richness of information.

POS Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings

PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give *up*
TO	to	*to* go, *to* him
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WRB	wh-abverb	where, when

Table 3.1: Penn Treebank Tagset

We extracted all the feature terms (excluding stop words) with their frequency from the POS tagged text, and calculated the importance of each of feature terms based on their probability distributions. Probability of the word *W* appearing in the input is calculated as *p(w)=n/N,* where n is the number of times the word appeared in the input, and *N* is the total number of feature terms in the input. Words having probability distribution of more than 0.0025 have been selected as feature terms and added to the previously available list of feature terms.

Term-Document Matrix: We computed term (Feature term) by document matrix from the input cluster of document. Table 3.2 shows the term document matrix for the example given in Figure 3.1 considering each sentence as a document. We treated each set of 2 to 3 sentences as a separate document while computing matrix so that we can compute the bi-gram score based on their occurrences at the sentence level.

	100	People	Sudanese	airbus	burst	Khartoum	airport
S1	1	1	1	1	1	0	0
S2	1	1	1	1	1	1	1
S3	1	0	1	1	1	1	1
S4	0	0	1	1	0	1	1

Table 3.2: Term-Document matrix for Figure 3.1

Matrix Decomposition: We computed the bi-gram scores for the pair of feature terms. We considered the feature terms with probabilities more than 0.0025 (lower threshold) for this phase. This removes the less important term thus drastically reducing the bi-gram matrix size and its computational time. Here bi-gram score means the occurrence of the bi-gram (two terms together) in the sentences. We computed bi-gram scores using Singular Value Decomposition (SVD) [35, 36].

The Singular Value Decomposition (SVD) of the term by document matrix X, If X is an m x n matrix then is

$$X=U\,S\,V^T$$

Where U is m x n with orthonormal columns, V is n X n with orthonormal Columns, and S is diagonal with the main diagonal entries sorted in decreasing order. A unique SVD feature is that it is capable of capturing and modeling interrelationships among terms so that it can semantically cluster terms and sentences. Furthermore, if a word combination pattern is salient and recurring in document, this pattern will be captured and represented by one of the singular vectors. The magnitude of the corresponding singular value indicates the importance degree of this pattern within the document. Any sentences containing this word combination pattern will be projected along this singular vector, and the sentence that best represents this pattern will have the largest index value with this vector. As each particular word combination pattern describes a certain topic/concept in the document, the facts described above naturally lead to the hypothesis that each singular vector represents a salient topic/concept of the document, and the magnitude of its corresponding singular value represents the degree of importance of the salient topic/concept.

Compute bi-gram Matrix: We compute the bi-gram matrix based on how often two terms co-occur. The term-document matrix X for a collection with n documents (paragraphs) and m terms is an m x n matrix with each column of the matrix representing a document. X can be viewed as a matrix with each row representing a term vector, i.e. a vector contacting the frequency of a term in each document. Thus similarity scores between terms can be calculated. A value representing the similarity t_{ij} between two terms I and j (with i , j ϵ {1,.......m}) is the dot product of the i-th and the j-th row of the term-document matrix, t_{ij} is nonzero if and only if a document exists, in which both terms I and j occur.

Let T be the square matrix containing all those similarity scores. T is called the bi-gram matrix and is:

$$\begin{aligned} T &= XX^T \\ &= USV^T (USV^T)^T \\ &= USV^TVSU^T \\ &= US^2U^T \\ &= (US)\,(US)^T \end{aligned}$$

We computed T matrix and set all the diagonal elements to zero's and then we normalized the matrix so that the sum of all the term-term scores is equal to 1. We call this normalized matrix as bi-gram score matrix. Table 3.3 shows the bi-gram matrix after decomposition for the term-document matrix shown in table 3.2.

3.2.3 Scoring & Summary Extraction

The final stage ranks the sentences based on the terms they contained and their bi-gram score, and finally the high ranking sentences are selected for inclusion in the summary. We assigned an importance weight to each sentence S_i in the input as a function of the importance of its bi-gram units BG_j

$$Weight\ (S_i) = F\ (p\ (BG_j))\ for\ all\ BG_j \in S_i$$

Different Summarizers can be obtained by making different choices for the composition function *F*. We considered *Product* of bigram scores as an appropriate candidate for assigning salience score to the sentences.

$$\text{for } (F{=}Product)\text{, Weight } (Si) = \pi_{BGj \in Si} P\ (BG_j)$$

Summary extraction is a three step procedure as discussed below.

1. **Sentence Score Computation:** We assigned an importance weight to each sentence S_i in the input as a function of the importance of its bi-gram units BG_j
   $Weight\ (S_i) = F\ (P\ (BG_j))\ for\ all\ BG_j \in S_i$
2. **Most salient Sentence:** Pick the best scoring sentences under the scoring function *F* from the above step.
3. **Repeat:** If desired summary length has not been reached, go back to step 1.

This chapter describes the proposed framework of the information extraction module and describe in details the process for extracting most salient sentences from the input set of source documents. The outcome of this module (an intermediate summary) consists of improperly ordered sentences which are given to the sentence ordering module as discussed in next chapter.

Chapter 4
Proposed Design for the Sentence Ordering Module in MDS

The problem of organizing information for multi document summarization so that the generated summary is coherent has received relatively little attention. While sentence ordering for single document summarization can be determined from the ordering of sentences in the input article, this is not the case for multi document summarization where summary sentences may be drawn from different input articles. The experiments [30] show that ordering significantly affects the reader's comprehension of a text. It shows that although there is no single ideal ordering of information, ordering is not an unconstrained problem; the number of good orderings for a given text is limited. In this chapter, a methodology for ordering the sentences based on sentence types is proposed. Even though the problem of ordering information for multi document summarization has received relatively little attention, we hypothesize that good ordering is crucial to produce good quality summaries.

4.1 A Study of Sentence Types

For the study of the sentence types we have chosen of the domain of scientific literatures. The scientific literatures generally follow a well defined structure, which makes it easy to identify sentence types. A scientific paper is a written report describing original research results [40]. The format of a scientific paper has been defined by centuries of developing tradition, editorial practice, scientific ethics and the interplay with printing and publishing services. The IMRaD [41] format has developed within the past 110 years and is the best choice for papers reporting laboratory studies. Figure 4.1 [40] shows the text organization of published articles in the British Medical Journal from 1935 to 1985 which shows that IMRad has become a common format for all the scientific documents. In the IMRaD format the text is structured in the following sequence:
Introduction: What question was studied and why?

Methods: How was the problem studied?

Results: What were the findings?

and

Discussion: What do these findings mean?

The IMRaD structure is a linear-analytic structure. The sequence of subtopics begins with an introduction to the issue or problem being studied and a review of what has been done so far. Then the method with which to approach the problem, the findings, and finally the conclusions and implications which are drawn from the findings are presented. A typical report structured according to the IMRaD-format would consist of the following parts: Title, Introduction, Method, Results, Discussion, Conclusions, Recommendations (optional), References, and Appendices (optional). In this dissertation five types of sentences; introduction, method, results, conclusions and scope or recommendation have been used for the ordering task. The information represented by each of these types of sentences is discussed below.

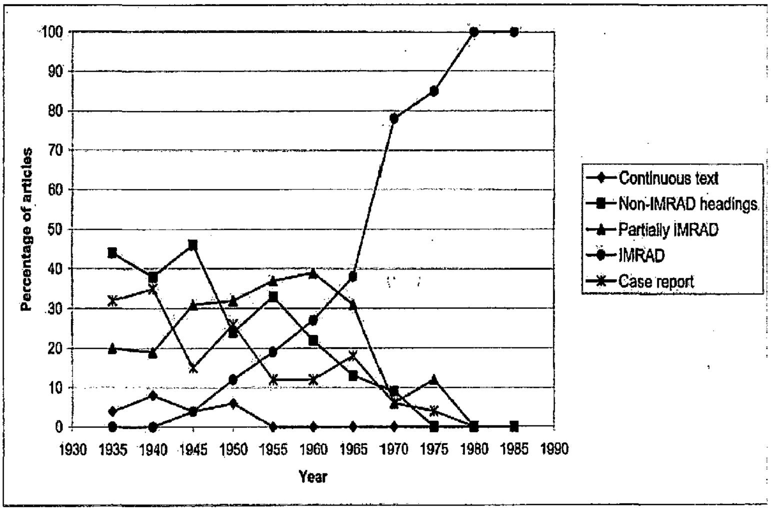

Figure 4.1: Text organization of published articles in the British Medical Journal

4.1.1 Introduction

The Introduction section begins by introducing the reader to the pertinent literature. An important function of the introduction is to establish the significance of the current work. The introduction sentences also describe the region in which the study was conducted. The introduction generally finishes with the statement of objectives or, with a brief statement of the principal findings. Either way, the reader must have an idea of where the paper is heading in order to follow the development of the evidence. The Introduction normally contains following types of sentences which tells about:

1. Nature and scope of the problem: What is the problem and why is it important to study?
2. Literature review: What has already been done? Review of the pertinent literature.

4.1.2 Methods

The main purpose of the Methods section is to provide enough detail for a competent worker to repeat current study and reproduce the results. The scientific method requires that results be reproducible. Usually these types of sentences describe the study site and climate in detail. Equipment and materials available off the shelf, sources of materials, measurements and errors of measurement are also described in this section. The Methods section gives full details of data collection, experimental design, sampling techniques, and so on. In a case study it is important to elaborate on the rationale for selection of the case to study. In the natural sciences this section is often called *Materials and Methods*. In the social sciences it is common to introduce a section called *Theory and Methods*, sometimes divided in two sections: *Theoretical Framework (Design)* and *Methods (Implementation)*.

4.1.3 Results

In the results section findings are presented. These sentences combine the use of text, tables and figures to condense data and highlight trends. In the discussion the results obtained from the application of proposed methods are discussed. Comparative studies between the findings to the findings of others or to expectations based on previous work are done in this section of sentences. The objectives of the study and to significance of the results in fulfilling them are also discussed.

4.1.4 Conclusion and Scope

The Conclusion section should discuss the overall study and the results and are found in the end of the document. These sentences discuss the principles, relationships, and generalizations shown by the result. The scope and recommendation section discuss the applications of the proposed methods and obtained results to the other domains. It also recommends certain set of guidelines for the people who wish to work in the same domain.

The above categorization of sentences is specifically designed for scientific documents. For other generic documents these types can be replaced by new types depending on the domain. For example in event based documents the method section is replaced by the event section, which describes about the whole event. The results section is replaced by the cause and effect section which tells about the cause and the aftereffects of the event.

4.2 Sentence Type Dictionaries

To identify the various different types of sentences we have generated separate dictionaries for each of the five types of sentences. The dictionary for any sentence type includes the most common and specific words, combination of words or phrases occurring in that particular type of sentences. For example the dictionary for results section's sentences includes *graphs, figures, performs 20% better etc.* words and the conclusion section's dictionary includes *we have described, the conclusion is, we find that etc.* words in it and likewise. These dictionaries are generated by extracting the most frequent words and phrases from a training set of scientific documents. In this dissertation three types of contents are added to the dictionaries: single words, pair of words and a maximum combination of three words. The single word dictionary is not enough to identify the difference among different types of sentences as the same word may be found in more than one type of sentences, for example the word *researches* can be found in introduction sentences as well as in methods, and results sentences. Therefore to make the difference more precise we have used combination of words and phrases.

4.3 The Framework of Sentence Ordering Module

The frame work for Sentence Ordering module is shown in Figure 4.2. This module is divided into two phases. First phase (Sentence Type Discovery) is to identify the types of the summary sentences and labeling them. The second phase (Ordering) deals with arranging the sentences in the correct order so as to get a coherent summary.

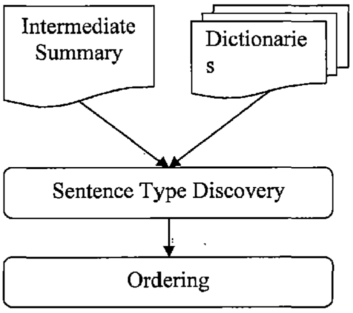

Figure 4.2: Sentence Ordering Module

4.3.1 Sentence Type Discovery

The first phase also known as the sentence type discovery phase is used to discover the types of the summary sentences. Figure 4.3 shows the block diagram for the sentence type discovery phase. The process of discovering the sentence type is completed in three steps. First step is preprocessing of the input summary, second step is identifying the sentence type and the third step is labeling the sentences with their types.

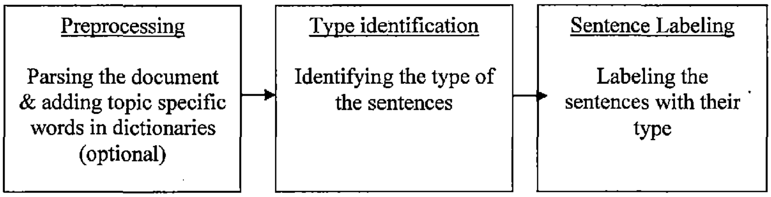
Preprocessing
Parsing the document & adding topic specific words in dictionaries (optional)
Type identification
Identifying the type of the sentences
Sentence Labeling
Labeling the sentences with their type

Figure 4.3: Framework for Sentence Type Discovery

The preprocessing includes identifying the sentences and words in the summary document. The preprocessing also adds the topic related words or combination of words into the dictionaries to make the ordering domain specific.

```
Begin:

  for Sentence S[i = 1 to Ns] {
      initialize S[i][1 to Nd] = 0;
      for Word W[j=1to Nw[i]] {
          for Dictionary D[k=1 to Nd] {
              if the D[k] contains(W[j] || W[j]+W[j+1] || W[j]+W[j+1]
              + W[j+2])
                  S[i][k] ++;
          }
          ST[i] = max(S[i][k=1 to Nd]);
      }
  }
End
```

Figure 4.4: Algorithm for Identification of Sentence Type

Once the dictionaries are ready and the summary document is parsed into the sentences the next task is to identify the types of the sentences. To identify sentence types we have used the algorithm shown in Figure 4.4. In this algorithm S[i] represent i^{th} sentence in the summary, W[j] represents *j*th word in the sentence and D[k] represents the k^{th} dictionary. Ns, Nw[i], and Nd represents the total no of sentences in the summary, total no of words in sentence S[i] and total no of dictionaries respectively. S[i][k] represents the probability (score) that the sentence S[i] falls under the category of the sentence type k. "W[j] +W[j+1]" represents the combination of the j^{th} and $(j+1)^{th}$ word.

The sentence labeling step labels each sentence to its type based on the words in it. This is done by comparing the type scores (S[i][k]) of a sentence S[i]. We label each sentence S[i] to its type ST[i] based on Where ST[i] is the maximum value out of all type scores (S[i][k]).

4.3.2 Ordering

The steps of the second phase of the sentence ordering module, known as Ordering phase are shown in Figure 4.5. This phase is divided in to two steps. First step is to determine the generic ordering for the type of the sentences present in the summary and the second task is to arrange the sentences in the summary according to this standard ordering.

Determining a generic ordering scheme can either be done manually or it can be derived from a set of training documents. In manual mode the human decides the generic order for the types of the sentences based on his knowledge and past experience. In the second case a set of sample documents are chosen and a set of ordering schemes are generated based on the sentences in these documents. Then the scheme which is found in majority of documents is selected as the standard ordering scheme. In this dissertation we have used the standard ordering scheme of the IMRaD format, which arranges the sentences in following sequence Introduction, Method, Results, Conclusion and Scope.

The first step of this phase is useful if we are ordering the sentences of other domains than the scientific documents.

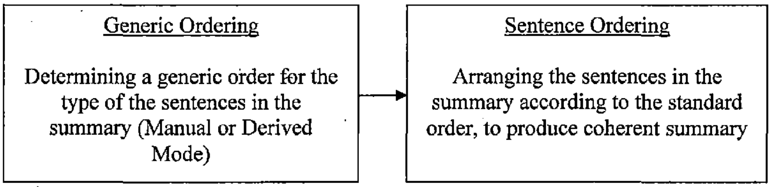
Generic Ordering
Determining a generic order for the type of the sentences in the summary (Manual or Derived Mode)
Sentence Ordering
Arranging the sentences in the summary according to the standard order, to produce coherent summary

Figure 4.5: Framework for Sentence Ordering phase

After the generic ordering scheme is selected the second task is to arrange the summary sentences in proper order. As we have already labeled the summary sentences with their respective types so it becomes easy to arrange them according to the standard order. The sentences which have the same sentence type are ordered according to following scheme. If the sentences are from the same source documents then the order in the source document is maintained. If the sentences are from different source documents then the sentences are arranged using chronological ordering algorithm. This whole process can be used in reverse manner also. In that case first the summary sentences are ordered as per the chronological ordering scheme and then the sentences from the same chronology can be reordered using the sentence type information.

This chapter discusses the second module of the proposed multi document summarization system which is used to order the summary sentences. The next chapter describes the summary evaluation methods used to evaluate the proposed summarization model and the results obtained from it.

Chapter 5
Results and Discussion

5.1 ROUGE: Evaluation Method for Automatic Summarization

Estimating the informativeness has been the focus of automatic summarization evaluation research. Various methods have been proposed to evaluate the summaries generated by multi document summarization systems. In this dissertation ROUGE [37] which stands for Recall-Oriented Understudy for Gisting Evaluation is used as the evaluation metric to determine the MDS performance. ROUGE is based on n-gram (window of size n words) co-occurrence between machine summaries and "ideal" human summaries. It includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans. ROUGE is currently the standard objective evaluation measure for the Document Understanding Conference [42]. ROUGE does not assume that there is a single "gold standard" summary. Instead it operates by matching the target summary against a set of reference summaries. ROUGE-1 through Rouge-4 are simple n-gram co-occurrence measures, which checks whether the n-gram (sequence of n words) in the reference summary is contained in the machine summary. ROUGE-L computes the longest common subsequence for the evaluation purpose. ROUGE-S uses the concept of skip-bigram. Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. Skip-bigram co-occurrence statistics measure the overlap of skip-bigrams between a candidate translation and a set of reference translations. ROUGE-SU is an extention of ROUGE-S which uses unigram as counting unit and computes the skip-bigram value for each pair of counting units. Lin (Lin and Hovy, 2003) has found that ROUGE-1 and ROUGE-2 correlate well with human judges.

ROUGE generates three scores (recall, precision and F-measure) for each evaluation. Recall is fraction of expert summary which is present in the summary generated by the system and is given as.

$$\text{Recall} = \frac{\textit{no of (sentences in system summary} \cap \textit{sentences in expert summary)}}{\textit{total no of sentences in expert summary}}$$

Precision is the fraction of the sentences extracted in the system summary that are present in the expert summary and is given as below.

$$\text{Precision} = \frac{\textit{no of (sentences in system summary} \cap \textit{sentences in expert summary)}}{\textit{total no of sentences in system summary}}$$

F-measure is the weighted harmonic mean of precision and recall. The general formula for F-measure is given as

$$F_\beta = \frac{(1+\beta 2)*\text{precision} * \text{recall}}{(\beta 2*\text{precision} + \text{recall})}$$

We have used the traditional (β=1) F-measure for our evaluation. This is also known as the F_1 measure, because recall and precision are evenly weighted. Previously, only one score is generated (recall). Precision and F-measure scores are useful when the target summary length is not enforced. We used ROUGE 1.5.5 to compute ROUGE-1, ROUGE-2 and ROUGE-SU automatic evaluation scores for the evaluation of our system.

5.2 Dataset Used for Validation

The performance of the proposed MDS summarizer has been validated using two publicly available datasets NIE and MDS [38]. Both the data sets consist of clusters of related news articles from. The NIE dataset consists of 48 clusters of news articles with an average of 8 articles per cluster. The MDS dataset consist of 6 clusters of news articles with an average of 2 documents each. The NIE dataset provide automatically clustered documents while the MDS dataset is clustered manually.

5.3 Results for Information Extraction Module

We used ROUGE 1.5.5 [44] for the evaluation of Information Extraction module. ROUGE provides a number of options to evaluate the summary obtained from the system against the expert summary. We have used following parameters of ROUGE 1.5.5, description of each parameter is given in table 5.1.

```
ROUGE-1.5.5.pl –n 4 -2 1 –u –c 95 –r 1000 -p 0.5 –f A –d
```

Option	Description
-2	Compute skip bi-gram co-occurrence, also specify the maximum gap length between two words (skip-bi-gram)
-u	Compute skip bi-gram as -2 but include unigram
-c	Specify CF% (0 <= CF <= 100) confidence interval to compute. The default is 95% (i.e. CF=95).
-d	Print per evaluation average score for each system.
-f	Select scoring formula: 'A' => model average (good for summarization task); 'B' => best model (good for machine translation)
-n	Compute ROUGE-N up to max-ngram length will be computed.
-p	Relative importance of recall and precision ROUGE scores. Alpha -> 1 favors precision, Alpha -> 0 favors recall.
-r	Specify the number of sampling point in bootstrap re-sampling (default is 1000). Smaller number will speed up the evaluation but less reliable confidence interval.

Table 5.1: ROUGE Options Used for Summary Evaluation

We present the results of our systems using different summary evaluation metrics of ROUGE method on the NIE dataset in Table 5.2. Figure 5.1 shows the graph for the recall values for ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-SU (R-SU). Figure 5.2

shows the graph for the precision values for R-1, R-2 and R-SU. Figure 5.3 shows the graph for the F-measure values for R-1, R-2 and R-SU.

Clusters	8	16	24	32	36	40	48
				ROUGE-1			
Recall	0.63327	0.61448	0.62452	0.62121	0.60638	0.6059	0.60541
Precision	0.46537	0.42667	0.42574	0.42192	0.41005	0.41002	0.41376
F-Measure	0.52803	0.49772	0.50022	0.4964	0.48357	0.48353	0.48664
				ROUGE-2			
Recall	0.3908	0.3596	0.35913	0.35891	0.33458	0.33279	0.32666
Precision	0.28157	0.24822	0.24298	0.24136	0.22429	0.22287	0.22101
F-Measure	0.322	0.29024	0.28614	0.28485	0.26522	0.26369	0.26079
				ROUGE-SU			
Recall	0.40619	0.38123	0.39312	0.38705	0.36727	0.36623	0.36544
Precision	0.22482	0.18909	0.18766	0.18272	0.17167	0.17136	0.17361
F-Measure	0.27354	0.24233	0.24357	0.23771	0.22448	0.22425	0.22722

Table 5.2: Recall, precision and F-score value for R-1, R-2 and R-SU

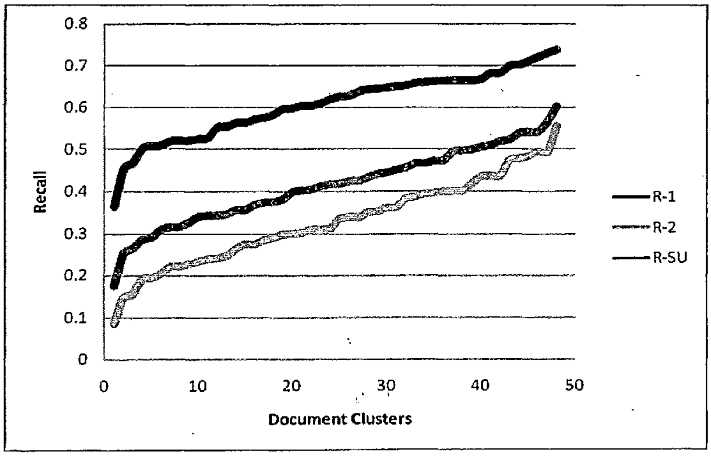

Figure 5.1: Recall Values for NIE-dataset at compression ratio 0.1

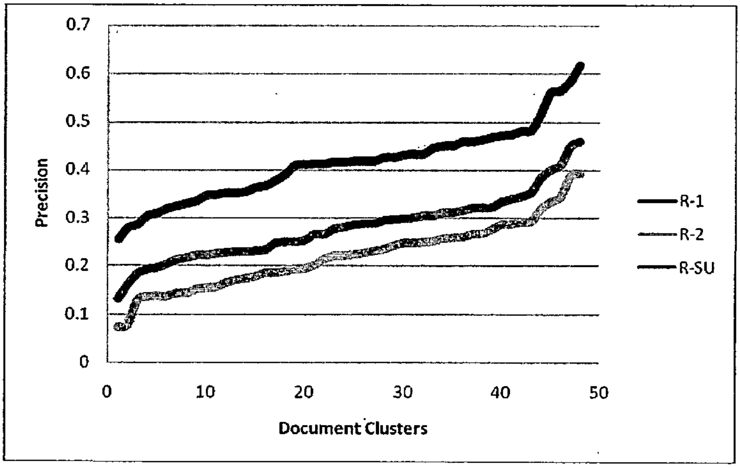

Figure 5.2: Precision Value for NIE-dataset at compression ratio 0.1

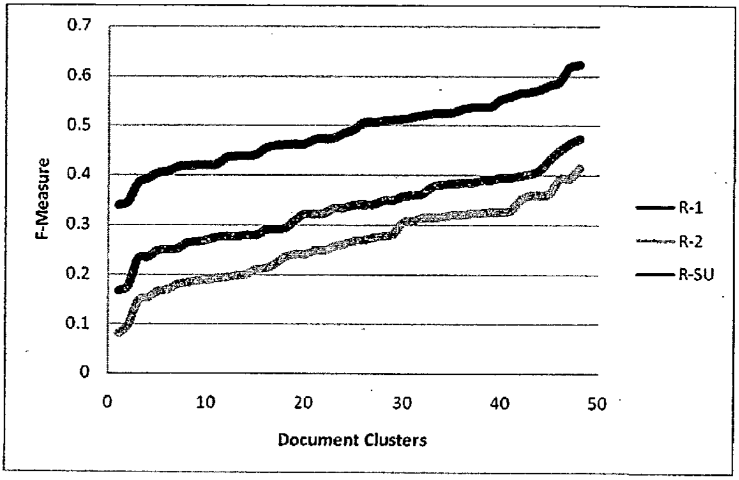

Figure 5.3: F-Measure Value for NIE dataset at compression ratio 0.1

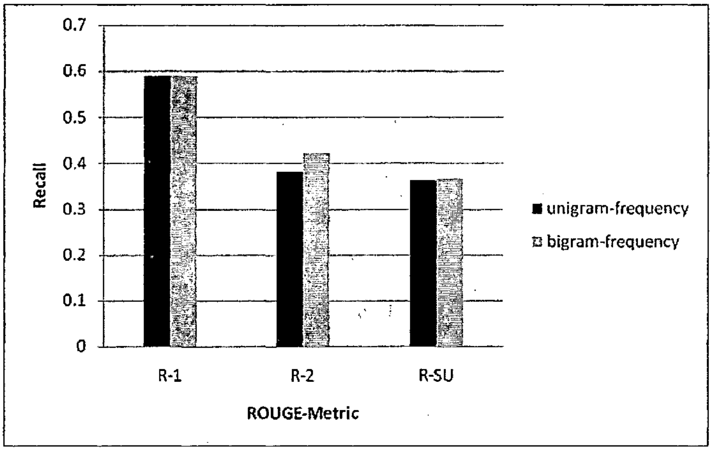

Figure 5.4: Recall Comparison for Proposed System with Unigram System

Figure 5.4, Figure 5.5 Figure 5.6 shows the comparison of recall, precision and f-measure between our system (bi-gram frequency) and the unigram system which uses the unigram (individual word) frequency for the information extraction task. Figure 5.4 show minimal improvement (R-1: 0.19%, R-2: 10.27% and R-SU: 0.85%) in the recall value from the unigram frequency system to bigram frequency system. The reason behind this is that ROUGE-1 uses single word co-occurrence for performance evaluation, still our system performs better (10% in case of R-2) which shows that more number of salient sentences present in expert summary are also present in the system summary than the unigram system. Figure 5.5 shows the comparison of precision values between both the systems. Here the proposed system (bigram system) significantly out-performs the unigram system (R-1: 28.93 %, R-2: 42.57%, R-SU: 64.62%). Here the high improvement in R-2 and R-SU shows that the sentences selected by the proposed system are the most salient sentences as they have high number of the bigram and skip-bigram co-occurrences with the expert summary. And the chances are high that same facts have been covered in both the summaries if we get high values for the bi-gram and skip bi-grams co-occurrences as discussed in chapter 3.

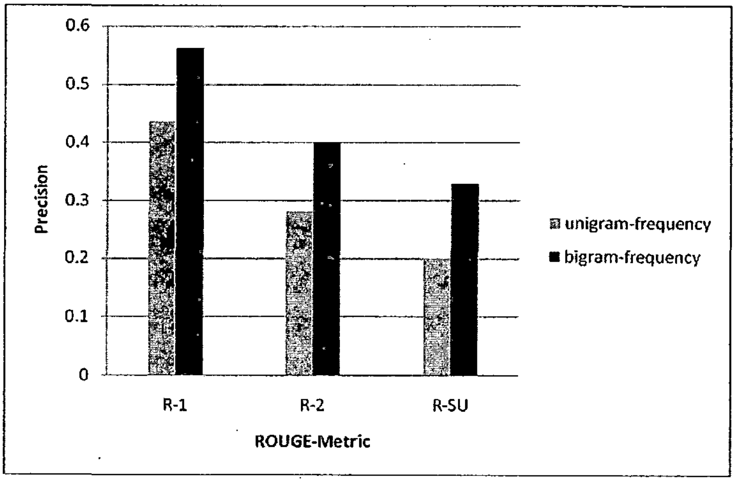

Figure 5.5: Precision Comparison for Proposed System with Unigram System

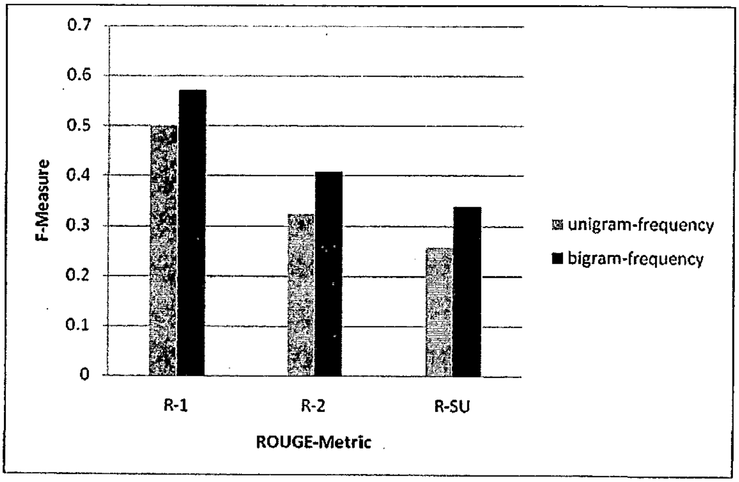

Figure 5.6: F-Measure Comparison for Proposed System with Unigram System

Figure 5.6 shows the overall comparison of the proposed system to the unigram system in terms of F-measure. The improvements gained are R-1: 14.44%, R-2: 26.48% and R-SU: 31.86%. These improvements show that our system works better than the unigram system and therefore co-relates well with the expert summaries.

Figure 5.7 shows the f-measure comparison of the proposed Information Extraction system with the previously existing centroid based summarizer MEAD [43] at the compression rate of 0.2 on the MDS dataset. MEAD uses three measures to score the sentences the unigram frequency, the position of the sentences and the overlapping with the first sentence (or the title). It is very clear from these figures that our system performs better than MEAD for the information extraction task (R-1: 27.14%, R-2: 10.12% and R-SU: 59.44%). It shows that the bi-gram feature is more important than the unigram feature as well as the location feature and title feature too.

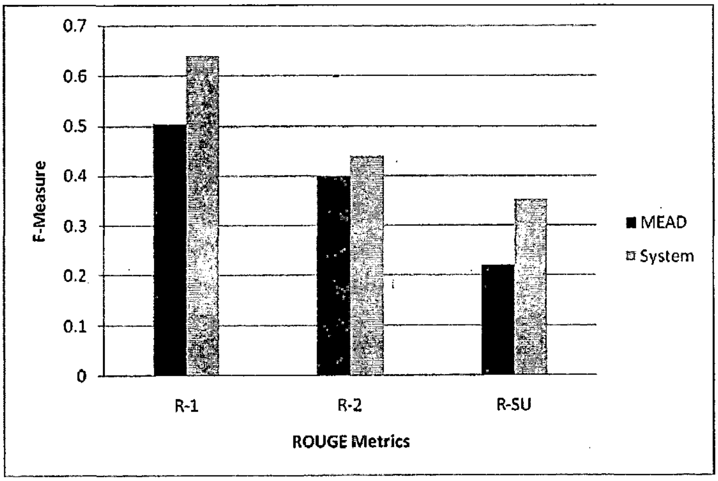

Figure 5.7: F-Measure Comparison for Proposed System and MEAD System

5.4 Results for Sentence Ordering Module

To evaluate the performance of the sentence ordering module we have used Kendall's tau (τ) coefficient [39] as the performance measure. Kendall's τ for the ordering task is evaluated as follows. Let $Y = y1 \ldots yn$ be a set of items to be ranked. Let π and σ denote two distinct orderings of *Y*, and *S* (π, σ) the minimum number of adjacent transpositions needed to bring π to σ. Kendall's τ is defined as:

$$\tau = 1 - \{2S\,(\pi,\,\sigma)\}/\,\{N\,(N - 1)/2\}$$

where *N* is the number of objects (i.e., items) being ranked. As can be seen, Kendall's τ is based on the number transpositions, that is, interchanges of consecutive elements, necessary to rearrange π into σ. In Table 5.3 the number of transpositions can be calculated by counting the number of intersections of the lines. The τ between the Reference and System 1 is 0.82, between the Reference and System 2 is 0.24, and between the two systems is 0.15. The metrics ranges from between -1 (inverse ranks) to 1 (identical ranks).

Kendall's τ seems particularly appropriate for the sentence-ordering tasks considered [39]. The metric is sensitive to the fact that some items may be always ordered next to each other even though their absolute orders might differ. It also penalizes inverse rankings. Comparison between the Reference and System 2 gives a τ of 0.24 even though the orders between the two models are identical modulo the beginning and the end. This seems appropriate given that flipping the introduction in a document with the conclusions seriously disrupts coherence.

	A	B	C	D	E	F	G	H	I	J
Reference	1	2	3	4	5	6	7	8	9	10
System 1	2	1	5	3	4	6	7	9	8	10
System 2	10	2	3	4	5	6	7	8	9	1

Table 5.3: Reference order and system orders for a text consisting of 10 items

Figure 5.8 shows the results of the proposed sentence ordering module based on the sentence types. We have validated the proposed sentence ordering technique on the summaries obtained from the NIE clusters and calculated the Kendall's tau (τ) coefficient for the ordering of each of these summaries against the ordering of the expert summaries for the same. We get an average Kendall's tau (τ) score of 0.76351, which is close to 1 and verifies that our system's sentence ordering 70 to 80 % identical as of the expert's sentence ordering.

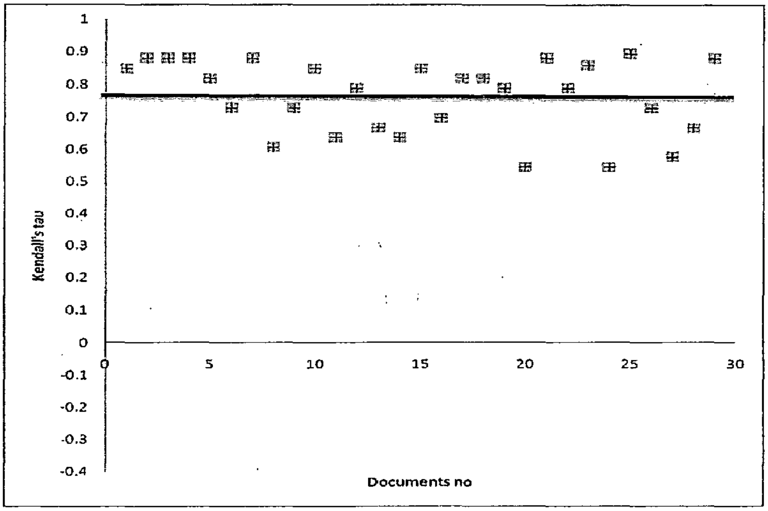

Figure 5.8: Kendall's tau (τ) coefficient Value for Ordered NIE Summaries

Next we show the result of the sentence type discovery module used in the proposed sentence ordering technique to identify the IMRaD structure discussed in chapter 4 of the scientific texts. Figure 5.9 (a) shows the no of sentences that has not been identified by the system. Figure 5.9 (b) shows the no of sentences identified as introductory sentences. Figure 5.9 (c) shows the no of sentences identified as method sentences. Figure 5.9 (d) shows the no of sentences identified as result sentences. Figure 5.9 (e) shows the no of sentences identified as conclusion sentences. Figure 5.9 (f) shows the no of sentences identified as scope and recommendation sentences.

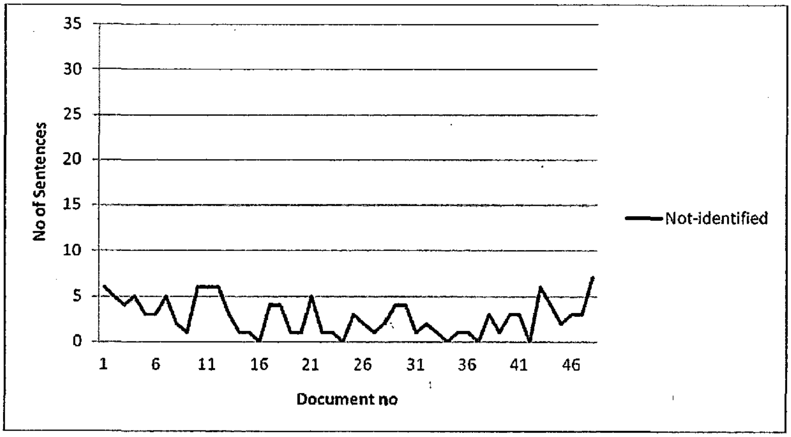

Figure 5.9 (a): No of Sentences Not Identified

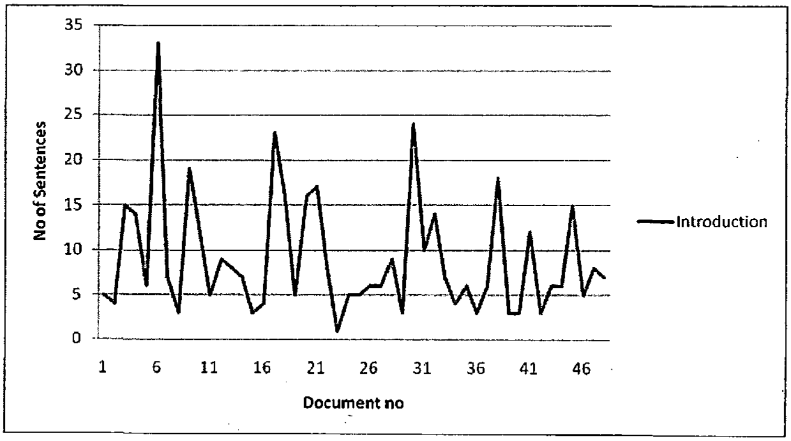

Figure 5.9 (b): No of Sentences Identified as Introduction Sentences

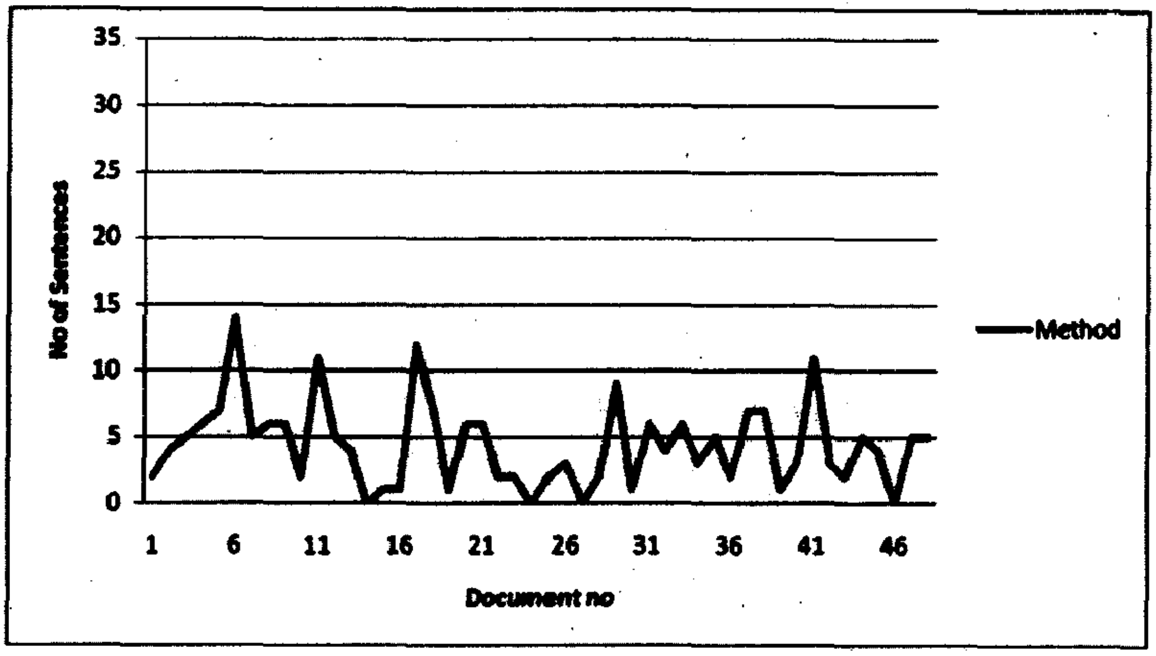

Figure 5.9 (c): No of Sentences Identified as Method Sentences

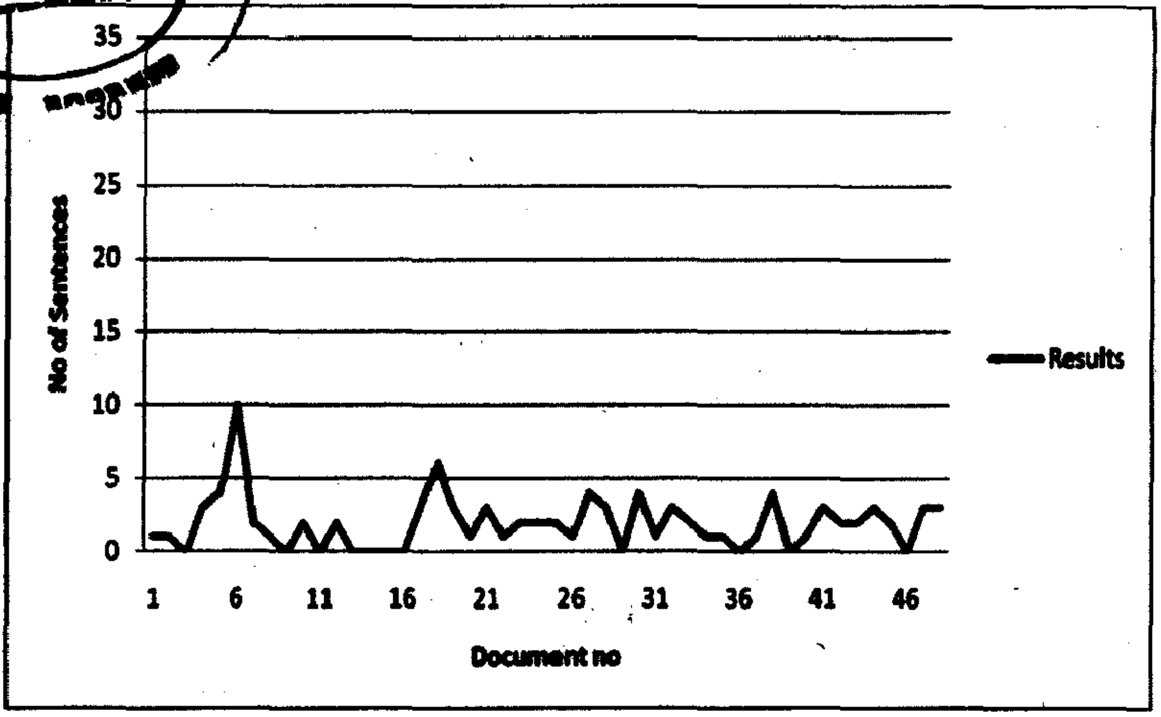

Figure 5.9 (d): No of Sentences Identified as Result Sentences

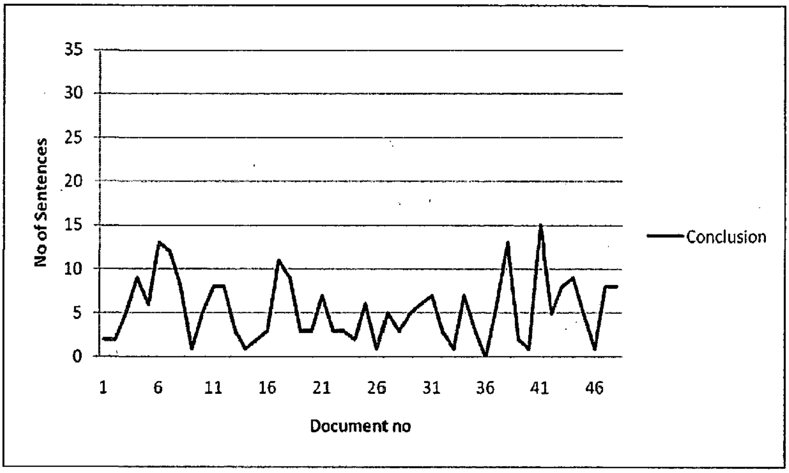

Figure 5.9 (e): No of Sentences Identified as Conclusion Sentences

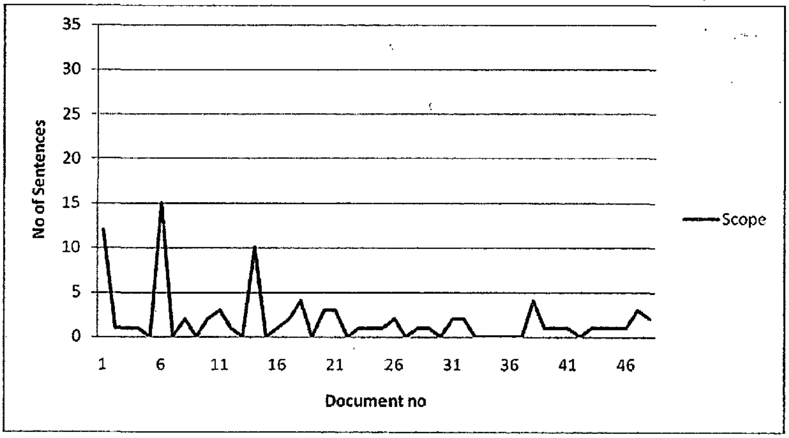

Figure 5.9 (f): No of Sentences Identified as Scope Sentences

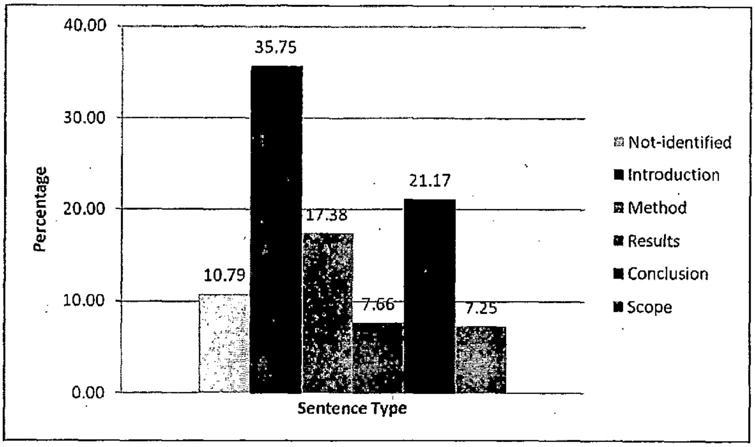

Figure 5.10: Percentage of Identified Sentence Types

Figure 5.10 shows the percentage of each type of identified sentences as well as the not identified sentences for the sentence type discovery module. The figure shows that our system identifies 90% of the sentences while 10% sentences are remained unidentified. This is due to the length and language of the sentences. Some sentences are very short and contain a language that is very specific to the domain. These types of the sentences are not identified by the system until we add these specific words to the sentence type dictionaries. Next we observe that the introduction type sentences have the most no of shares in the summary (~36%). This shows that these type of sentences are very important to be included in a summary they introduce the reader about the topic, which is very important from the user point of view. The next highest percentage of identified sentence type is the conclusion sentences (~21%). This shows that they are the second choice for the inclusion in summary. This is because they contain sentences which summarize the overall event or document and tell about the findings which are the common focus of the user after reading the introduction. Next is the method type (~17 %) which tells about the overall event or process and which the user reads after he finds the introduction and conclusion interesting to him. The results (~7%) and the scope (~7%)

are generally the sentences which the user likes to reads in the full articles rather than in summary as the main results are included in conclusion part, so these types of sentences are found less in summaries. These observations are quite useful from the summary extraction point of view and conform to the normal behavior of the user.

With this chapter, the effectiveness of the proposed techniques information extraction and sentence ordering has been demonstrated. The results of both the phases of the proposed model have been illustrated. The comparison of the proposed bigram system with the existing unigram system, MEAD system and a Kendall's tau value of 0.76 shows that the proposed system works significantly better in extracting information and ordering the sentences to give a coherent summary.

Chapter 6
Conclusion and Future Scope

6.1 Conclusion

In this dissertation we have proposed techniques for the information extraction and sentence ordering tasks in multi document summarization. Information extraction task deals with extracting the most salient sentences from the source documents for the inclusion in summary while the sentence ordering task deals with the representation of these sentences to form a coherent summary. The comparison of the proposed system to the traditional system is given in table 6.1 for both the tasks.

Summarization Task	Traditional Systems	Proposed System
Information Extraction	Unigram (Individual word) frequency	Bi-gram (window of size 2 words) frequency
Sentence Ordering	Chronological ordering, dependency network (graph) based ordering	Chronological ordering augmented with sentence type based ordering

Table 6.1: Comparison of Features between the Traditional Systems and Proposed System

The proposed idea for the information extraction is based on the bi-gram (window of size two important words) frequency rather than the traditional unigram frequency. We observed that the sequence of important words conveys more meaning than the individual words. If a sequence of important words in the system extracted summary co-occurs with a sequence in the expert summary than it is more probable that both the sequences contains same information, while if individual words co-occur there are less chances that same facts have been covered in system generated summary and the expert summary. The high precision value gained by our system over the unigram frequency system confirms this observation. Overall the proposed information extraction module performs 14% better (F-Measure with equal weightage to recall and precision) than the information extraction systems that make use of unigram frequency.

The idea behind the sentence ordering is based on the types of sentences that are generally found in scientific texts. We have used IMRaD structure for the study of such types of sentences in this dissertation. The proposed system gives sentence orderings which are 70 to 80% identical as the sentence orderings given by the experts. We observed an important fact during this step about the nature of the summary sentences. We find that the first choice for the summary sentences is the sentences which introduce the event, process or document topic to the user. The next choice for summary sentences is the conclusion sentences then follows the method sentences. The result and the scope sentences are not that much important from summary point of view. These observations can be used in the information extraction task.

6.2 Future Scope

The work can be extended in following ways:

- Discovering new ideas or methods to generate semantically rich n-grams units which are more meaningful than the bi-gram unit. The problem with this can be that such n-grams units (n>2) are not so frequent. This makes the extraction task more difficult.

- The sentence ordering task is limited to only scientific documents, it can be extended in other domains. To extend it to other domains we only need to find the types of sentences that frequent in the documents of that particular domain.

REFERENCES

[1]. P. Lyman, H. R. Varian, K. Swearingen, P. Charles, N. Good, L.L. Jordan, J. Pal. "How Much Information 2003?". School of Information Management and Systems, University of California at Berkeley, 2003. Available at http://www.sims.berkeley.edu/how-much-info-2003.

[2]. Mani, I. 2001. "Automatic Summarization". John Benjamin's Publishing Co., limited edition available at http://books.google.co.in/books?id=WVUfl1JsKVQC &printsec=frontcover&dq=automatic+summarization&sig=MBOJj4C4nU75yHQu A52mrm1rMKs

[3]. Udo Hahn, Inderjeet Mani. "The Challenges of Automatic Summarization". *Computer, vol. 33, no. 11, pp. 29-36*, Nov., 2000

[4]. Salton, G., Singhal, A., Mitra, M., and Buckley, C. 1997. "Automatic text structuring and summarization". *Information Processing and Management: an International Journal. 33*, 2 (Mar. 1997), 193-207.

[5]. Lin, C. and Hovy, E. 2001. "From single to multi-document summarization: a prototype system and its evaluation". *In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (Philadelphia, Pennsylvania, July 07-12, 2002).* Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 457-464.

[6]. G. C. Stein, A. Bagga, and G. B. Wise. "Multi-document summarization: Methodologies and evaluations". *In Proceedings of the 7th Conference on Automatic Natural Language Processing (TALN '00), pages 337--346*, October 2000

[7]. Moens M.-F., Angheluta R., Dumortier J. "Generic technologies for single- And multi-document summarization," *Information Processing and Management, 41 (3), pp. 569-586*, 2005

[8]. Radev, D. R., Hovy, E., and McKeown, K. 2002. "Introduction to the special issue on summarization". *Comput. Linguist. 28*, 4 (Dec. 2002), 399-408

[9]. Barzilay, R., McKeown, K. R., and Elhadad, M. 1999. "Information fusion in the context of multi-document summarization". *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics (College Park, Maryland, June 20 - 26, 1999).* Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 550-557

[10]. Radev, D. R. and McKeown, K. R. 1998. "Generating natural language summaries from multiple on-line sources". *Comput. Linguist. 24*, 3 (Sep. 1998)

[11]. Luhn, H. P. 1969. "The Automatic Creation of Literature Abstracts". *IBM Journal of Research and Development 2(2)*, 1969

[12]. H. P. Edmundson. "New Methods in Automatic Extracting". *Journal of the ACM (JACM), v.16 n.2, p.264-285,* April 1969

[13]. Goldstein, J., Mittal, V., Carbonell, J., and Kantrowitz, M. 2000. "Multi-document summarization by sentence extraction". *In NAACL-ANLP 2000 Workshop on Automatic Summarization - Volume 4 (Seattle, Washington, April 30 - 30, 2000).* ANLP/NAACL Workshops. Association for Computational Linguistics, Morristown, NJ, 40-48

[14]. Goldstein, J., Mittal, V., Carbonell, J., and Callan, J. 2000. "Creating and evaluating multi-document sentence extract summaries". *In Proceedings of the Ninth international Conference on information and Knowledge Management (McLean, Virginia, United States, November 06 - 11, 2000).* CIKM '00. ACM, New York, NY, 165-172.

[15]. Schiffman, B., Nenkova, A., and McKeown, K. 2002. "Experiments in multi document summarization". *In Proceedings of the Second international Conference on Human Language Technology Research (San Diego, California, March 24 - 27, 2002).* Human Language Technology Conference. Morgan Kaufmann Publishers, San Francisco, CA, 52-58.

[16]. Knight, K. and Marcu, D. 2002. "Summarization beyond sentence extraction: a probabilistic approach to sentence compression". *Artificial Intelligence 139, 1 (Jul. 2002), 91-107.*

[17]. D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Celebi, E. Drabek, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel and Z. Zhang, "MEAD - a platform for multidocument multilingual text summarization," *in LREC 2004*, 2004.

[18]. Kupiec, J., Pedersen, J., and Chen, F. 1995. "A trainable document summarizer". *In Proceedings of the 18th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Seattle, Washington, United States, July 09 - 13, 1995).* E. A. Fox, P. Ingwersen, and R. Fidel, Eds. SIGIR '95. ACM, New York, NY, 68-73

[19]. Radev, D. R., Jing, H., Styś, M., and Tam, D. 2004. "Centroid-based summarization of multiple documents". *Information Processing and Management. 40, 6 (Nov. 2004), 919-938.*

[20]. Yeh, J., Ke, H., Yang, W., and Meng, I. 2005. "Text summarization using a trainable summarizer and latent semantic analysis". *Information Processing and Management. 41, 1 (Jan. 2005), 75-95.*

[21]. Mihalcea, R. 2004. "Graph-based ranking algorithms for sentence extraction, applied to text summarization". *In Proceedings of the ACL 2004 on interactive Poster and Demonstration Sessions (Barcelona, Spain, July 21 - 26, 2004).* Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 20.

[22]. G Erkan, DR Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization", *Journal of Artificial Intelligence Research,* 2004

[23]. C. Lin and E. Hovy. 2002. "NeATS in DUC 2002". *In Proceedings of the DUC 2002 Workshop on Multi-Document Summarization Evaluation,* Philadelphia, PA, July.

[24]. R Barzilay, N Elhadad, KR McKeown, "Inferring Strategies for Sentence Ordering in Multidocument News Summarization", *Journal of Artificial Intelligence Research,* 2002

[25]. Lapata, M. 2003. "Probabilistic text structuring: experiments with sentence

ordering". *In Proceedings of the 41st Annual Meeting on Association For Computational Linguistics - Volume 1 (Sapporo, Japan, July 07 - 12, 2003).* Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 545-552.

[26]. Bollegala, D., Okazaki, N., and Ishizuka, M. 2006. "A bottom-up approach to sentence ordering for multi-document summarization". *In Proceedings of the 21st international Conference on Computational Linguistics and the 44th Annual Meeting of the ACL (Sydney, Australia, July 17 - 18, 2006).* Annual Meeting of the ACL. Association for Computational Linguistics, Morristown, NJ, 385-392.

[27]. Okazaki, N., Matsuo, Y., and Ishizuka, M. 2004. Improving chronological sentence ordering by precedence relation. *In Proceedings of the 20th international Conference on Computational Linguistics (Geneva, Switzerland, August 23 - 27, 2004).* International Conference On Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, 750.

[28]. Barzilay, R. and Lapata, M. 2008. "Modeling local coherence: An entity-based approach". *Comput. Linguist. 34*, 1 (Mar. 2008), 1-34.

[29]. R. Barzilay and L. Lee. "Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization". *In HLT-NAACL 2004: Proceedings of the Main Conference, pages 113--120,* 2004.

[30]. Barzilay, R., Elhadad, N., and McKeown, K. R. 2001. "Sentence ordering in multi document summarization". *In Proceedings of the First international Conference on Human Language Technology Research (San Diego, March 18 - 21, 2001).* Human Language Technology Conference. Association for Computational Linguistics, Morristown, NJ, 1-7.

[31]. Yang-Wendy Wang, "Sentence Ordering for Multi Document Summarization in Response to Multiple Queries". A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the School of Computing Science, Simon Fraser University, 2006.

[32]. Radev, D., Otterbacher, J., Winkel, A., and Blair-Goldensohn, S. 2005.

"NewsInEssence: summarizing online news topics". *Commun. ACM 48*, 10 (Oct. 2005), 95-98.

[33]. A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*, 2005.

[34]. Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. 2003." Feature-rich part-of-speech tagging with a cyclic dependency network". *In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1 (Edmonton, Canada, May 27 – June 01, 2003).* North American Chapter Of The Association For Computational Linguistics. Association for Computational Linguistics, Morristown, NJ, 173-180.

[35]. J Steinberger, K Jezek, "Text Summarization and Singular Value Decomposition", Advances In Information Systems: *Third International conference , ADVIS,* 2004

[36]. Qin Bing, Liu Ting, Zhang Yu, Li Sheng, "Research on multi-document summarization based on latent semantic indexing " *Journal of Harbin Institute of Technology (New Series). Vol. 12, no. 1, pp. 91-94.* Feb. 2005

[37]. Lin, C. Y. 2004. "ROUGE: A Package for Automatic Evaluation of Summaries". *In Proceedings of the Workshop on Text Summarization Branches Out, post-conference workshop of ACL 2004,* Barcelona, Spain.

[38]. MDS and NIE (clusters of related news article) datasets, www.tangra.si.umich.edu/clair/CSTBank

[39]. Lapata, M. 2006. "Automatic Evaluation of Information Ordering: Kendall's" Tau. *Comput. Linguist. 32, 4 (Dec. 2006), 471-484.*

[40]. Szklo, Moyses. "Quality of scientific articles. Rev. Saúde Pública [online]". 2006, *vol. 40, no. spe [cited 2008-06-21], pp. 30-35.Scientific Literature structure*

[41]. Sollaci LB, Pereira MG. "The introduction, methods, results, and discussion (IMRAD) structure: a fifty-year survey". *J Med Libr Assoc. 2004;92:364-7*

[42]. Document Understanding Conference (DUC) publication list, *http://www-*

nlpir.nist.gov/projects/duc/pubs.html

[43]. The online MEAD demo, *http://tangra.si.umich.edu/clair/md/demo.cgi*

[44]. Download link for ROUGE *http://www.isi.edu/licensed-sw/see/rouge/ index.html*

[45]. Download link for Stanford POS-tagger *http://nlp.stanford.edu/software/ tagger.shtml*

LIST OF PUBLICATIONS

[1] **Amit Gusain**, R. C. Joshi "Multi Document Summarization: Information Extraction and Sentence Ordering," *Journal of Advances in Computational Sciences and Technology (ACST)*, (paper under review).

[2] **Amit Gusain**, Surbhi Bhasin, "A Novel Approach for Sentence Ordering in Multi document Summarization based on Sentence Types," 4th *International Conference on Challenges and Development in IT (ICCDIT-2008) organized by Punjab College of Technical Education, May 2008*. (in press)

APPENDIX A:

SUMMARY GENERATED BY PROPOSED SYSTEM, TRADITIONAL SYSTEM AND HUMAN EXPERT

1. Summary generated by Proposed (bi-gram) System

Moscow Mayor Yuri Luzhkov visited the blast site and said a terrorist act" appeared to be the most likely cause of the explosion on the southeast fringes of the capital, near the Moscow River. Other officials said the blast was caused by a gas leak or explosives possibly stored in the building -- which could bring the mayor under fire for negligence in enforcing building regulations. Sergei Bogdanov, press officer of the Moscow branch of the security service, said the blast may have been caused by explosives stored in the building. An anonymous caller told the Interfax news agency that the Moscow explosion and a Saturday night bomb blast in southern Russia were in response to Russia's military campaign against Islamic rebels in the southern territory of Dagestan. MOSCOW, Sept(AFP) - Russia's Federal Security Service (FSB) said a massive bomb had destroyed a Moscow apartment block in which at least people died and scores were wounded on Thursday, the third such attack to terrorize Russia in days. An anonymous caller to Interfax said the blast and a car-bomb earlier this week at a military apartment building in Dagestan were "our response to the bombing of villages in Chechnya and Dagestan." Emergency officials expressed hope that more survivors would be found even hours after the blast, while investigators said they would work through the night in the hunt for clues, news reports said.

2. Summary generated by Traditional (unigram) System

MOSCOW (AP) -- Authorities searched for survivors and clues after a massive explosion tore apart a nine-story apartment building today, killing at leastpeople and leaving dozens more feared dead. Moscow Mayor Yuri Luzhkov visited the blast site and said a terrorist act" appeared to be the most likely cause of the explosion on the southeast fringes of the capital, near the Moscow River. It was almost like something flew overhead and then exploded. Moscow has experienced natural gas explosions in the past because of the city's crumbling infrastructure, including a blast in Julythat killed six people. MOSCOW, Sept(AFP) - Russia's Federal Security Service (FSB) said a massive bomb had destroyed a Moscow apartment block in which at leastpeople died and scores were wounded on Thursday, the third such attack to terrorise Russia indays. "If it is confirmed that this is a terrorist act, and everything is leading that way, we shall have to acknowledge that the echo of war in Dagestan is sounding in Moscow," said Yury Luzhkov, Moscow's mayor. "Every minute counts. The federal authorities together with the Moscow city authorities, are already doing and will do all that is necessary to help the victims," Yeltsin said. They are all dead," she said.

3. Summary generated by Human Expert

MOSCOW (AP) -- Authorities searched for survivors and clues after a massive explosion tore apart a nine-story apartment building today, killing at least 23 people and leaving dozens more feared dead. Moscow Mayor Yuri Luzhkov visited the blast site and said a terrorist act" appeared to be the most likely cause of the explosion on the southeast fringes of the capital, near the Moscow River. Other officials said the blast was caused by a gas leak or explosives possibly stored in the building -- which could bring the mayor under fire for negligence in enforcing building regulations. An anonymous caller told the Interfax news agency that the Moscow explosion and a Saturday night bomb blast in southern Russia were in response to Russia's military campaign against Islamic rebels in the southern territory of Dagestan. MOSCOW, Sept 9 (AFP) - Russia's Federal Security Service (FSB) said a massive bomb had destroyed a Moscow apartment block in which at least 34 people died and scores were wounded on Thursday, the third such attack to terrorise Russia in 10 days. 2 Itar-Tass news agency, quoting an official of the emergencies ministry, said at least 34 people were killed and that there was no hope of finding survivors. 3 senior officials surmised that the blast was connected to the month-long Islamic rebellion in the Caucasus republic of Dagestan.

APPENDIX B:
SOURCE CODE

FeatureTermExtraction.java

```
import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;
class FeatureTermExtraction{
public static void main(String args[]){
try{
int lines=0,no_of_words=0;
Hashtable FeatureTerms = new Hashtable();
Hashtable StopWords= new Hashtable();
Object frequency;
int i=0;
String str, str1, tmp, pos_tag;
StringTokenizer s_tokenizer;
FileReader f_name= new FileReader("../Data/input/KeyFeatures/featureterms_file.txt");
BufferedReader b_reader= new BufferedReader(f_name);
while((str=b_reader.readLine())!=null)
{
i++;
s_tokenizer=new StringTokenizer(str.trim(),"\n");
tmp=s_tokenizer.nextToken();
FeatureTerms.put(tmp, new Integer(0));
}
b_reader.close();
i=0;
FileReader f_name1= new FileReader("../Data/input/StopWords/english.stop");
BufferedReader b_reader1= new BufferedReader(f_name1);
while((str=b_reader1.readLine())!=null)
{
i++;
s_tokenizer=new StringTokenizer(str.trim(),"\n");
tmp=s_tokenizer.nextToken();
System.out.println(tmp);
StopWords.put(tmp, new Integer(0));
}
System.out.println("Total Stop Words " + i);
b_reader1.close();
i=0;
String dirname="../Data/intermediate/postagged";
File dir = new File(dirname);
```

```
if(dir.isDirectory())
{
String s[]=dir.list();
for(int file_iter=0; file_iter<s.length; file_iter++)
{
File file = new File(dirname+"/"+s[file_iter]);
FileWriter fout = new FileWriter("../Data/input/KeyFeatures/"+s[file_iter]);
if(!file.isDirectory())
{
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
while((str=b_Reader.readLine())!=null)
{
str=str.toLowerCase();
s_tokenizer= new StringTokenizer(str, " ");
while(s_tokenizer.hasMoreTokens())
{
str1=s_tokenizer.nextToken().trim();
no_of_words++;
i=str1.indexOf("/");
if(i > -1){
str=str1.substring(0,i);
pos_tag=str1.substring(i+1);
if(!StopWords.contains(str) && (pos_tag.startsWith("vb")||pos_tag.startsWith("nn")))
{
if((frequency=FeatureTerms.get(str))!=null)
FeatureTerms.put(str,new Integer(((Integer)frequency).intValue()+1));
else
FeatureTerms.put(str,new Integer(0));
}
}
}
}
Vector key_set= new Vector(FeatureTerms.keySet());
Object array[]= new Object[key_set.size()];
key_set.copyInto(array);
Arrays.sort(array);
for(i=0;i<array.length; i++)
{
str=(String)array[i];
if((((Integer)(frequency=FeatureTerms.get(str))).intValue()<(no_of_words*.0025))
{
FeatureTerms.remove(str);
}
else
{
```

```
fout.write(str+"\n");
FeatureTerms.put(str, new Integer(0));
}
}
no_of_words=0;
b_Reader.close();
}
fout.close();
}
}
 }catch(Exception e){System.out.println(e);}
}
}
```

BigramMatrix.java

```
import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;
class BigramMatrix
{
public static void main(String args[])
{
try
{
int lines=0;
Hashtable words = new Hashtable();
int columns=0;
Object frequency;
int i=0;
String str,str1, tmp,pos_tag;
StringTokenizer s_tokenizer;
i=0;
String dirname="../Data/intermediate/postagged";
String outdir="../Data/intermediate/TermDocumetMatrix";
File dir = new File(dirname);
if(dir.isDirectory())
{
String s[]=dir.list();
for(int file_iter=0; file_iter<s.length; file_iter++)
{
File file = new File(dirname+"/"+s[file_iter]);
FileWriter fout = new FileWriter(outdir+"/"+s[file_iter]);
FileReader f_name= new FileReader("../Data/input/KeyFeatures/"+s[file_iter]);
```

```
BufferedReader b_reader= new BufferedReader(f_name);
while((str=b_reader.readLine())!=null)
{
s_tokenizer=new StringTokenizer(str.trim(),"\t");
tmp=s_tokenizer.nextToken();
words.put(tmp, new Integer(0));
}
b_reader.close();
if(!file.isDirectory())
{
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
while((str=b_Reader.readLine())!=null)
{
str=str.toLowerCase();
s_tokenizer= new StringTokenizer(str, " ");
while(s_tokenizer.hasMoreTokens())
{
str1=s_tokenizer.nextToken().trim();
i=str1.indexOf("/");
if(i > -1){
str=str1.substring(0,i);
pos_tag=str1.substring(i+1);
if(str.length()<3)
continue;
else if(pos_tag.startsWith("vb")||pos_tag.startsWith("jj")||pos_tag.startsWith("nn"))
{
if((frequency=words.get(str))!=null)
words.put(str,new Integer(((Integer)frequency).intValue()+1));
else
words.put(str,new Integer(0));
}
}
}
Vector key_set= new Vector(words.keySet());
Object array[]= new Object[key_set.size()];
key_set.copyInto(array);
Arrays.sort(array);
for(i=0;i<array.length; i++)
{
str=(String)array[i];
if(lines==0)
{
columns=array.length;
}
if((((Integer)(frequency=words.get(str))).intValue())>0)
{
words.put(str, new Integer(0));
}
fout.write((Integer)frequency+",");
}
fout.write("\n");
lines++;
}
b_Reader.close();
System.out.println("file " + file.getAbsolutePath() + " lines "+lines);
}
fout.close();
mat_decomposition(lines, columns, s[file_iter]);
words.clear();lines = 0; columns= 0;
}
}
}catch(Exception e){System.out.println(e);}
}
static void mat_decomposition(int row, int col, String fname)
{
try
{
FileReader f_name= new
FileReader("../Data/intermediate/TermDocumetMatrix/"+fname);
BufferedReader b_Reader= new BufferedReader(f_name);
String str;
double[][]mat= new double[row][col];
StringTokenizer s_tokenizer;
int i=0, j=0;
while((str=b_Reader.readLine())!=null)
{
s_tokenizer=new StringTokenizer(str.trim(),",");
for(j=0;j<col;j++)
{
mat[i][j]=Integer.parseInt(s_tokenizer.nextToken().trim());
}
i++;
}
b_Reader.close();
System.out.println("shoulld be"+row+" "+col);
System.out.println("present be"+i+" "+col);
FileOutputStream fs= new
FileOutputStream("../Data/intermediate/BigramMatrix/"+fname);
PrintWriter pw= new PrintWriter(fs, true);

Matrix A= new Matrix(mat);
```

```
SingularValueDecomposition svd= A.svd();
Matrix U=svd.getU();
Matrix V=svd.getV();
Matrix S= svd.getS();
Matrix X= U.times(S);
Matrix T= X.times(X.transpose());
pw.println(T.getRowDimension());
T.print(pw,2,1);
pw.close();
}catch(Exception e){System.out.println("in function mat_decomposition " + e);}
}
}
```

<u>SummaryExtraction.java</u>

```
import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;
class BigramFrequencyComparator implements Comparator{
Hashtable word_tokens;
Hashtable word_index;
double[][]bigram_matrix;
BigramFrequencyComparator(Hashtable wrds, Hashtable wrdindex, double[][]
bg_matrix){
word_tokens= wrds;
word_index=wrdindex;
bigram_matrix=bg_matrix;
}
public int compare(Object a, Object b){
double val= value(a)-value(b);
if(val>0)
return 1;
if(val==0)
return 1;
return (-1);
}
double value(Object a){
String str=(String) a;
str= str.toLowerCase();
Object tokvalue;
int[] indices= new int[150];
double sum=0, product=1, value,avg;
int no_of_tokens=0, vi=0;
StringTokenizer s_tokenizer= new StringTokenizer(str,"_-@$%!#&*+,()[]`\'=;:/\n");
while(s_tokenizer.hasMoreTokens()){
str= s_tokenizer.nextToken().trim();
if((tokvalue=word_tokens.get(str))!=null){
no_of_tokens++;
value=((Double)tokvalue).doubleValue();
sum+=value;
if(value!=0)
product*=value;
}
if((tokvalue=word_index.get(str))!=null){
indices[vi++]=((Integer)tokvalue).intValue();
}
}
if(product==1)
product=0;
avg=sum/no_of_tokens;
double relative_sum=0, relative_product=1, relative_value, relative_avg;
int row_no, col_no, relative_no_of_tokens=0;
for(int i=0; i<vi;i++){
for(int j=0; j <vi; j++){
row_no=indices[i];
col_no=indices[j];
relative_no_of_tokens++;
relative_sum+=bigram_matrix[row_no][col_no]/(j-i);
if(bigram_matrix[row_no][col_no]!=0)
relative_product*=bigram_matrix[row_no][col_no]/(j-i);
}
}
if(relative_product==1.0)
relative_product=0;
relative_avg=relative_sum/relative_no_of_tokens;
double alpha=.3, beta=.7;
return (alpha*product + beta*relative_product);
}
}
class OrderComparator implements Comparator{
Hashtable Sent_list;
OrderComparator(Hashtable s_list){
Sent_list =s_list;
}
public int compare(Object a, Object b){
System.out.println( (Integer)Sent_list.get(a) + " " + (Integer)Sent_list.get(b));
if(((Integer)Sent_list.get(a)).intValue() > ((Integer)Sent_list.get(b)).intValue())
return 1;
return -1;
}
```

```
}
class SummaryExtraction{
String directory;
Hashtable word_tokens;
Hashtable Sents;
Hashtable word_index;
double [][]bigram_matrix;
Hashtable StopWord_tokens;
HashSet allsentences;
int total_sentences, abs_Summary_Size;
int sent_no;

SummaryExtraction(){
word_tokens= new Hashtable();
Sents= new Hashtable();
word_index = new Hashtable();
StopWord_tokens = new Hashtable();
directory="../Data/input/RawText1"; //directory for source data set
allsentences = new HashSet();
total_sentences=0;
sent_no=0;
}
public void Initialize(){
try{
Object frequency;
int i=0, freq=0, totalwords=0;
double p_writer;
String str,str1,pos_tag, dirname;
StringTokenizer s_tokenizer;
FileReader f_name = new FileReader("../Data/input/StopWords/english.stop"); //stop
word list
BufferedReader b_reader = new BufferedReader(f_name);
while((str=b_reader.readLine())!=null){
str.trim();
StopWords.put(str,new Integer(0));
}
b_reader.close();
                    dirname="../Data/intermediate/postagged";// postagged files
FileWriter fout= new FileWriter("../Data/intermediate/noun_vb_jj.txt");
File dir= new File(dirname);
String s[]=dir.list();
for(int file_iter=0; fi<s.length;fi++){
File file = new File(dirname+"/"+s[fi]);
if(!file.isDirectory()){
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
while((str=b_Reader.readLine())!=null){
total_sentences++;
str=str.toLowerCase();
s_tokenizer=new StringTokenizer(str," ");
while(s_tokenizer.hasMoreTokens()){
str1=s_tokenizer.nextToken().trim();
i=str1.lastIndexOf("/");
if(i!=-1){
str = str1.substring(0,i);
pos_tag=str1.substring(i+1);
if(str.length()<3||StopWords.get(str)!=null)
continue;
else if(pos_tag.startsWith("vb")||pos_tag.startsWith("nn")||pos_tag.startsWith("jj"))
{
totalwords++;
if((frequency=word_tokens.get(str))==null)
word_tokens.put(str, new Integer(0));
else
word_tokens.put(str, new Integer(((Integer)frequency).intValue()+1));
}
}
}
}
b_Reader.close();
}
Enumeration keys_set= word_tokens.keys();
while(keys_set.hasMoreElements()){
str=(String)keys_set.nextElement();
frequency= word_tokens.get(str);
freq=((Integer)frequency).intValue();
p_writer=((double)freq)/totalwords;
word_tokens.put(str, new Double(p_writer));
if((int)(p_writer*1000)!=0)
fout.write(str+"\t"+freq+"\n");
}
initialize_word_index(s[file_iter]);
initialize_bigram_matrix(s[file_iter]);
storeSentences(s[file_iter]);
FileWriter fout1= new FileWriter("../Data/output/Summary-pro/nie-"+s[file_iter]);
BigramFrequencyComparator cmp= new
BigramFrequencyComparator(this.word_tokens,this.word_index,this.bigram_matrix);

Vector Sentences = new Vector();
Vector sortedSentences = new Vector();

Sentences.addAll(0, this.allsentences);
```

```
Collections.sort(Sentences, cmp);
Collections.reverse(Sentences);

abs_Summary_Size = (10 * total_sentences)/100;
System.out.println(s[file_iter] +"\t" + total_sentences +"\t "+ abs_Summary_Size);

Iterator sortiter = Sentences.iterator();
while (sortiter.hasNext()) {
if (abs_Summary_Size == 0) sortiter.next();
else {
sortedSentences.add(sortiter.next());
abs_Summary_Size--;
}
}
OrderComparator cmpr= new OrderComparator(Sents);
Collections.sort(sortedSentences, cmpr);
Iterator iter = sortedSentences.iterator();
while (iter.hasNext()) fout1.write(iter.next() + "\n");
fout1.close();
word_tokens.clear();word_index.clear(); allsentences.clear();Sents.clear();
total_sentences=0;abs_Summary_Size=0;sent_no=0;
}
}catch(Exception e){e.printStackTrace();}
}
public void initialize_bigram_matrix(String fname)throws Exception{
String str;
StringTokenizer s_tokenizer;
FileReader f_name = new FileReader("../Data/intermediate/BigramMatrix/"+fname);
BufferedReader b_Reader= new BufferedReader(f_name);
int i=0, j=0;
int sz= Integer.parseInt(b_Reader.readLine().trim());
b_Reader.readLine();
bigram_matrix= new double[sz][sz];
double tot=0;
while((str=b_Reader.readLine())!=null){
s_tokenizer=new StringTokenizer(str.trim(),"\t");
if(!s_tokenizer.hasMoreElements())
break;
for(j=0;j<sz; j++){
bigram_matrix[i][j]=Double.parseDouble(s_tokenizer.nextToken().trim());
if(i!=j)
tot+= bigram_matrix[i][j];
else
bigram_matrix[i][j]=0;
}
i++;
```

```
}
b_Reader.close();
for(i=0;i<sz;i++)
for(j=0;j<sz;j++)
bigram_matrix[i][j]/=tot;
}
public void initialize_word_index(String fname)throws Exception{
String str, tmp;
StringTokenizer s_tokenizer;
FileReader f_name= new FileReader("../Data/input/KeyFeatures/"+fname);//imp features
BufferedReader b_reader=new BufferedReader(f_name);
int i=0;
while((str=b_reader.readLine())!=null){
s_tokenizer=new StringTokenizer(str.trim(), "\n");
tmp= s_tokenizer.nextToken().trim();
tmp=tmp.toLowerCase();
word_index.put(tmp, new Integer(i++));
}
b_reader.close();
}
public void storeSentences(String fname)
{
try{
String str, dirname=directory;
File file = new File(dirname+"/"+fname);
if(!file.isDirectory())
{
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
while((str=b_Reader.readLine())!=null)
{
storeSnt(str);
}
b_Reader.close();
}
}catch(Exception e){e.printStackTrace();}
}
public void storeSnt1(String str){
allsentences.add(str);
Sents.put(str, new Integer(sent_no++));
}
public void storeSnt(String str){
int index=0;
while(str!=null){
index=0;
do{
```

```
index=str.indexOf(". ",index+1);
if(index<50 && index>0)
index = str.indexOf(".",index+1);
if(index<0 || index>(str.length()-10))
break;
}while(!(Character.isUpperCase(str.charAt(index+3))||Character.isUpperCase(str.charAt(
index+2))));
if(index>0){
allsentences.add(str.substring(0,index));
Sents.put(str.substring(0,index), new Integer(sent_no++));
str= str.substring(index+1);
}
else if(str!=null){
allsentences.add(str);
Sents.put(str, new Integer(sent_no++));
str=null;
}
}
}
public static void main(String args[]){
try{
Summary docs= new Summary();
docs.Initialize();
System.out.println("store Sentences over");
BigramFrequencyComparator cmp= new
BigramFrequencyComparator(docs.word_tokens,docs.word_index,docs.bigram_matrix);
Vector Sentences = new Vector();
Vector sortedSentences = new Vector();
Sentences.addAll(docs.allsentences);
Collections.sort(Sentences, cmp);
Collections.reverse(Sentences);
Iterator sortiter = Sentences.iterator();
abs_Summary_Size = (10 * total_sentences)/100;
System.out.println(total_sentences +"\t "+ abs_Summary_Size);
while (sortiter.hasNext()) {
if (abs_Summary_Size == 0) sortiter.next();
else {
sortedSentences.add(sortiter.next());
abs_Summary_Size--;
}
}
Iterator iter = sortedSentences.iterator();
while (iter.hasNext()) fout.write(iter.next() + "\n");
fout.close();
System.out.println("Summary Sentences Extracted");
}catch(Exception e){System.out.println("3rd "+e);}
```

```
}
}
```

Dictionary.java

```
import java.io.*;
import java.io.File;
import java.util.*;
public class Dictionary {
private final int nrWords = 1000;
private Hashtable dict[];
public Dictionary() {
dict  = new Hashtable[10];
dict[0] = new Hashtable(nrWords);
dict[1] = new Hashtable(nrWords);
dict[2] = new Hashtable(nrWords);
dict[3] = new Hashtable(nrWords);
dict[4] = new Hashtable(nrWords);
dict[5] = new Hashtable(nrWords);
dict[6] = new Hashtable(nrWords);
dict[7] = new Hashtable(nrWords);
dict[8] = new Hashtable(nrWords);
dict[9] = new Hashtable(nrWords);
add();
}
public boolean contains(String phrase, int dictNo) {
if(dict[dictNo].containsKey((phrase.toLowerCase()).trim())){
return true;
}
return false;
}
public void add(){
try{
String dirname="Dictionaries";
File dir= new File(dirname);
String s[]=dir.list();
for(int file_iter=0; file_iter<s.length;file_iter++){
File file = new File(dirname+"/"+s[file_iter]);
if(!file.isDirectory()){
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
String str;
while((str=b_Reader.readLine())!=null){
String commentstart = new String("//");
String emptystring = new String("");
if (!str.startsWith(commentstart) && !str.equals(emptystring)) {
dict[file_iter].put((str.toLowerCase()).trim(), new Integer(1));
}
}
b_Reader.close();
}
}
}catch(Exception e){System.out.println(e+" Dictionary");};
}
}
```

GetDictionaryWords.java

```
import java.io.*;
import java.io.File;
import java.util.StringTokenizer;
import java.util.*;
class GetDictionaryWords
{
public static void main(String args[])
{
try
{
int lines=0,no_of_words=0;
Hashtable FeatureTerms = new Hashtable();
Hashtable StopWords= new Hashtable();
int columns=0;
Object frequency;
int i=0;
String str,str1, tmp,pos_tag,first,second,third;
first=""; second="";third="";
StringTokenizer s_tokenizer;
FileReader f_name1= new FileReader("../Data/input/StopWords/english.stop");
BufferedReader b_reader1= new BufferedReader(f_name1);
while((str=b_reader1.readLine())!=null)
{
s_tokenizer=new StringTokenizer(str.trim(),"\n");
tmp=s_tokenizer.nextToken();
StopWords.put(tmp.trim(), new Integer(0));
}
b_reader1.close();
String dirname="../Data/intermediate/Dictionary/Training";
File dir = new File(dirname);
if(dir.isDirectory())
{
```

```
String s[]=dir.list();
for(int file_iter=0; file_iter<s.length; file_iter++)
{
File file = new File(dirname+"/"+s[file_iter]);
FileWriter fout = new FileWriter("../Data/intermediate/Dictionary/"+s[file_iter]);
if(!file.isDirectory())
{
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);
while((str=b_Reader.readLine())!=null)
{
str=str.toLowerCase();
s_tokenizer= new StringTokenizer(str, " -%!&+,()[]\'=;:/\n");
if(no_of_words==0){
first=s_tokenizer.nextToken().trim();
second=s_tokenizer.nextToken().trim();
}
while(s_tokenizer.hasMoreTokens())
{
third=s_tokenizer.nextToken().trim();
str1=first;
if(str1.trim()!="."){
no_of_words++;
if(str1.length()>2 && StopWords.get(str1)==null){
if( (frequency=FeatureTerms.get(str1))!=null)
FeatureTerms.put(str1,new Integer(((Integer)frequency).intValue()+1));
else
FeatureTerms.put(str1,new Integer(0));
}
str1=first+" " +second;
if((frequency=FeatureTerms.get(str1))!=null)
FeatureTerms.put(str1,new Integer(((Integer)frequency).intValue()+1));
else
FeatureTerms.put(str1,new Integer(0));
str1=first+" " +second+" "+third;
if((frequency=FeatureTerms.get(str1))!=null)
FeatureTerms.put(str1,new Integer(((Integer)frequency).intValue()+1));
else
FeatureTerms.put(str1,new Integer(0));
}
first=second;
second=third;
}
}
Vector key_set= new Vector(FeatureTerms.keySet());
Object array[]= new Object[key_set.size()];
key_set.copyInto(array);
Arrays.sort(array);
for(i=0;i<array.length; i++)
{
str=(String)array[i];
if((((Integer)(frequency=FeatureTerms.get(str))).intValue())>no_of_words*.0001)
fout.write(str+"\n");
}
b_Reader.close();
}

FeatureTerms.clear();no_of_words=0;
fout.close();
}
}
 }catch(Exception e){e.printStackTrace();}
}
}
```

Ordercomparator.java

```
import java.util.*;
public class OrderComparator implements Comparator {
public OrderComparator(){
}
public int compare(Object o1, Object o2) {
double o1SentWgt = ((Sentence)o1).getSentenceType();
double o2SentWgt = ((Sentence)o2).getSentenceType();

if (o1SentWgt > o2SentWgt) return 1;
else if (o1SentWgt < o2SentWgt) return -1;
else return 0;
}
}
```

Sentence.java

```
import java.util.*;
public class Sentence{
private VectortheSentence;
private Stringsent;
private int  sentenceType;
public int  typeScore[];
public Sentence(String sentence) {
```

```
typeScore= new int[10];
theSentence = new Vector();
String[] words = sentence.split("\\s|,|;|:|\\t");
for (int i = 0; i < words.length; i++) {
if (words[i].length() != 0) {
theSentence.add(new Word(words[i].trim()));
}
}
sent = sentence;
calcType();
}
public void calcType()
{
Word curword, bi_word, tri_word;
String first, second, third="";
int max= -1;
ListIterator listiter = theSentence.listIterator();
first = ((Word)listiter.next()).getWord();
second = ((Word)listiter.next()).getWord();
while (listiter.hasNext()) {
third =((Word)listiter.next()).getWord();
curword = new Word(first);
bi_word = new Word(first + " "+ second);
tri_word = new Word(first +" "+second + " " + third);
for(int i=0; i<10; i++){
if(curword.isType(i) || bi_word.isType(i) || tri_word.isType(i))
{
typeScore[i]++;
if(typeScore[i]>=max){
max=typeScore[i];
sentenceType=i;
}
}
}
first = second;
second = third;
}
for(int i=0; i<10; i++){
if((new Word(first)).isType(i) || (new Word(second)).isType(i) || (new Word(first+ " " +
second)).isType(i))
{
typeScore[i]++;
if(typeScore[i]>=max){
max=typeScore[i];
sentenceType=i;
}
}
}
}
public int getSentenceType() {
return sentenceType;
}
public String getSentenceString() {
return sent;
}
public Vector getWords() {
return theSentence;
}
}
```

SentenceOrdering.java

```
import java.io.*;
import java.util.*;
class SentenceOrdering
{
static Vector sentences;
public static void orderSentences(){
try{
FileWriter fout= new FileWriter("../Data/output/test/single-word/Final-A.spl");
ListIterator listiter = sentences.listIterator();
Collections.sort(sentences, new OrderComparator());
while (listiter.hasNext()) {
Sentence s1= (Sentence)listiter.next();
fout.write(s1.getSentenceString() + "." + s1.getSentenceType() + "\n");
}
fout.close();
}catch(Exception e){System.out.println(e+"SentenceOrdering 2");}
}
public static void main(String args[]) {
String sent;
try
{
sentences = new Vector();
String dirname="../Data/output/test/single-word/A.sent";
File file= new File(dirname);
FileReader f_reader= new FileReader(file.getAbsolutePath());
BufferedReader b_Reader= new BufferedReader(f_reader);int i=0;
while((sent=b_Reader.readLine())!=null){
sentences.add(new Sentence(sent));
}
}catch(Exception e){System.out.println(e +" SentenceOrdering:main  ");}
orderSentences();
}
}
```

Word.java

```
public class Word {
private String theWord;
private boolean type[];
static Dictionary dicts = new Dictionary();;
public Word(String word) {
type = new boolean[10];
theWord = word;
calcType();
}
public void calcType() {
for(int i=0; i<10; i++){
type[i]=false;
if(dicts.contains(theWord, i)){
type[i]=true;
}
}
}
boolean isType(int i){
return type[i];
}
public String getWord() {
return theWord;
}
}
```