Automatic Text Summarization using Graph based Compression and Rule based Modification

A thesis submitted in partial fulfillment of the requirement for the Degree of

Master of Computer Science and Engineering

Jadavpur University

By

Shouvik Roy

Registration No.: 129007 of 2014-15

Examination Roll No.: M4CSE1619

Under the Guidance of

Dr. Sudip Kumar Naskar

Department of Computer Science and Engineering

Jadavpur University, Kolkata-700032

India

2016

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Recommendation

This is to certify that the dissertation entitled "Automatic Text Summarization using Graph based Compression and Rule based Modification" has been carried out by Shouvik Roy (University Registration No.: 129007 of 2014-15, Examination Roll No.: M4CSE1619) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Computer Science and Engineering. The research results presented in the thesis have not been included in any other thesis submitted for the award of any degree in any other University or Institute.

……………………………………………………

Dr. Sudip Kumar Naskar (Thesis Supervisor)

Department of Computer Science and Engineering

Jadavpur University, Kolkata-32

Countersigned

……………………………………………………

Prof. Debesh Kumar Das

Head, Department of Computer Science and Engineering,

Jadavpur University, Kolkata-32.

……………………………………………………

Prof. Sivaji Bandyopadhyay

Dean, Faculty of Engineering and Technology,

Jadavpur University, Kolkata-32.

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

Certificate of Approval*

This is to certify that the thesis entitled "Automatic Text Summarization using Graph based Compression and Rule based Modification" is a bona-fide record of work carried out by Shouvik Roy in partial fulfillment of the requirements for the award of the degree of Master of Computer Science and Engineering in the Department of Computer Science and Engineering, Jadavpur University during the period of June 2015 to May 2016. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the thesis only for the purpose for which it has been submitted.

…………………………………………………………………………….

Signature of Examiner 1

Date:

…………………………………………………………………………….

Signature of Examiner 2

Date:

*Only in case the thesis is approved

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

<u>Declaration of Originality and Compliance of Academic Ethics</u>

I hereby declare that this thesis entitled "Automatic Text Summarization using Graph based Compression and Rule based Modification" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Computer Science & Engineering.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Shouvik Roy

Registration No: 129007 of 2014-15

Exam Roll No.: M4CSE1619

Thesis Title: Automatic Text Summarization using Graph based Compression and Rule based Modification

…………………………………………..

Signature with Date

Acknowledgement

I would like to express my deepest gratitude to my advisor, **Dr. Sudip Kumar Naskar**, Assistant Professor, Department of Computer Science and Engineering, Jadavpur University for his admirable guidance, care, patience and for providing me with an excellent atmosphere for doing research. Our numerous scientific discussions and his many constructive comments have greatly improved this work.

Words cannot express my indebtedness to **Prof. Sivaji Bandyopadhyay**, Department of Computer Science and Engineering, Jadavpur University for his amazing guidance and supervision. I am deeply grateful to him for the long discussions that helped to enrich the technical content of this manuscript. Without his enthusiasm, encouragement, support and continuous optimism this thesis would hardly have been completed.

This thesis would not have been completed without the inspiration and support of a number of wonderful individuals including my parents and friends— my thanks and appreciation to all of them for being part of this journey and making this thesis possible.

…………………………………………..

Shouvik Roy

Registration No: 129007 of 2014-15

Exam Roll No.: M4CSE1619

Department of Computer Science & Engineering

Jadavpur University

Content

List of Figures

List of Tables

1. Introduction

Building natural summaries automatically from text document(s) has remained a tough challenge for a long time. The need for such systems is growing due to the large amount of data available at our disposal as well as the redundancy present in those data. Text summarization effectively curtails all the redundancy from text document(s) and presents the user with an informative output. Radev, Hovy and McKeown (2002) define a summary as a text that is produced from one or more texts that contain a significant portion of the information in the original text, and that is no longer than half of the original text(s). There are many techniques devised so far which can generate summaries from a given document. The two most popular methods among summarization lie in extraction and abstraction. However in our work here, we opt to take a middle ground between them and use a compression based algorithm which will incorporate features from both extractive and abstractive summarization techniques. Many text-to-text generation procedures (e.g., Jing (2000), Clark and Lapata (2008)) involving sentence compression and abstraction operate on relatively short input file ranging from one sentence to a few paragraphs. We have come a long way from summarizing a few lines to large document(s). In the next sections we will discuss briefly the basics of summarization and provide detailed instances how the domain of summarization have progressed from basic feature extractive methods to advanced phrase merging and sentence fusion As we delve into the thesis all the major aspects and relevant techniques of summarization will be described in details.

1.1 Automatic Text Summarization

Text Summarization is the process of obtaining the most important and salient information from a given document(s) to produce an abridged version of the original document(s). It must be done so in a manner that the resultant summary must be syntactically and semantically coherent while at the same time being devoid of redundant and useless information. Important kinds of summaries that are the focus of current research include outlines of document, abstracts on scientific articles, headlines of news articles, snippets of web page, and answers to complex questions constructed by summarizing multiple documents. The task of text summarization can be achieved using a wide range of approaches, from domain-specific template-based methods relying on information extraction (White et al., 2001) to fully abstractive text-to-text generation, an approach that holds great expectations but that is still at an early stage (Genest and Lapalme, 2012). Neto et al. (2002) incorporated machine learning algorithms for feature extraction to perform summarization tasks. Another machine learning based approach was taken by Kan and McKeown (2002) where they induced preference among content planning using annotated corpus for training. Some techniques also consider the relation between sentences or the discourse structure by using synonyms of the words or anaphora resolution (Mani and Bloedorn, 1997). However our work focuses more on a compression based summarization technique. Somewhere between the two extremes of extraction and abstraction lies another approach which consists in modifying the source text sentences in order to create alternative sentences that are either shorter, or that combine information found in different sentences. Shortening sentences is known as sentence compression, and has been successfully used to improve extractive systems (Gillick and Favre, 2009). By compressing sentences, via temporal clauses removal or deletion of unnecessary phrases, more sentences, and hopefully more information, can be added to the summary (Knight and Marcu, 2002; Galley and McKeown, 2007). Combining different sentences to create a more informative sentence is known as sentence fusion. Fusing sentences allows creating a new sentence that regroups information spread across different source sentences, and can improve in many ways a summary (e.g. reducing redundancy while improving coherence and information coverage). One way to fuse sentences is to use their dependency parse trees and align their branches before generating a new sentence from the fused parse tree, a process known as linearization. However, creating a sentence from the fusion of the parse trees is difficult and often leads to ungrammatical sentences (Filippova and Strube, 2008). Barzilay and McKeown (2005) whose work is seminal in the field of summarization were among the first to introduce a competitive multi-document summarization system based on sentence fusion technique where they merged phrases based on content relevancy. After clustering related sentences into themes, they fuse the dependency parse trees of sentences in each cluster and generate sentences, ultimately selecting the best fusion via scoring against a language model. Some researchers like Knight and Marcu (2000) used statistical approaches for the tree modification where they considered sentence reduction a translation process using a noisy channel model. Another method for sentence fusion that does not rely on external resources has been introduced by (Filippova, 2010). Her approach consists in using a word graph of the sentences to be fused, and choosing a path in the graph that keeps the common information while providing a new sentence. This work was later extended by (Boudin and Morin, 2013) to generate more informative sentences by re-ranking fusion candidates according to the keyphrases they contain. Other sentence compression techniques used to incorporate grammatical compression and syntactic modifications include work by Galley and McKeown (2007) who used lexicalized markov grammar for sentence compression. Their method relies on a head-driven markovization of SCFG compression rules. Cheung and Penn (2014) created summaries combining dependency subtrees using distributional semantics. They use a novel sentence enhancement technique which extends sentence fusion by combining subtrees of many. A recent work by Bing et al. (2015) works purely on a syntactic front by merging important phrases of maximum salience. They first calculate phrase salience score by extracting the NPs and VPs from the parse tree and assigning a score to them. This is followed by compatibility relation check using resolution rules (Lee at al., 2013). Then the phrase-based content optimization is done using some statistical means. Sentence fusion and abstraction is a difficult task in itself, and its feasibility has been questioned (Daume III and Marcu, 2004), however, its promising results make it an interesting domain despite the difficulties to evaluate it intrinsically (Thadani and McKeown, 2013).

1.2 Different Types of Summarization

1.2.1 Single Document versus Multi Document Summarization

One of the basic distinguishing parameter among different types of summarization is based upon the input texts. The summary can be generated from a single input document as well as multiple input documents. Single document summarization is mostly used for generating simple headline outputs, outlines of a text or snippet generation. For multi-document summarization, the primary objective is to condense all the text available in the documents and to generate a summary that encompasses the relevant information present in all the documents. This is mostly used for summarizing related news article or web content pertaining to similar topics.

1.2.2 Extractive versus Abstractive Summarization

The most widely used distinguishing feature is the manner in which the summarization process is implemented. An extractive summarizer works on the selection and verbatim inclusion of “material” from the source document(s) in the summary; the “material” usually being sentences, paragraphs or even phrases. Extraction based text summarization extracts relevant objects from the given input document(s), without modifying the objects themselves; it merely copies the information deemed most important by the system to the summary. On the other hand, abstraction based summarization does not rely on simple extraction to generate summary; instead it does so by forming new sentences intuitively from the given document(s).It involves the identification of the most salient concepts prevalent in the source document(s), the fusion and the appropriate presentation of those instances.

1.2.3 Query Independent versus Query Dependent Summarization

A generic summarization is one which does not consider a particular user or information while generating summaries. The summary simply produces information in a domain non-specific manner. Such summarization is called query independent summarization since there is no external query or domain specific knowledge provided for the summary to focus on. On the other hand, query specific systems try to create a summary of the information found in the document(s), which is relevant to a user query. In a sense, we can say that the query-oriented summarization systems are user-focused; adapting each time to the explicitly expressed needs of the users

1.2.4 Text versus Multimedia Summarization

The medium used to represent the content of the input document(s) can be used as a parameter as well. Thus, we can have text, or multimedia (e.g. image, speech, and video apart from textual content) summarization. The most studied case is, of course, text summarization. However, there are also summarization systems that deal, for example, with the summarization of broadcast news and of diagrams.

1.3 A Basic Architectural Overview

The practices of automatic summarization vary widely across many dimensions, including source length, summary length, style, source, topic, language, and structure. Most typical are summaries of a single news document down to a headline or short summary, or of a collection of news documents down to a headline or short summary. A few researchers have focused on other aspects of summarization, including single sentence (Knight and Marcu, 2002), paragraph or short document (Daume III and Marcu, 2002), query-focused (Mittal and Berger, 2000), or speech (Hori and Furui, 2003). The primary research challenge in developing an efficient summarization technique lies in two areas: identification of the fragments conveying relevant information and combination of the fragments into a sentence.

The techniques relevant to, and the challenges faced in each of these tasks can be quite different. Nevertheless, they all rely on one critical assumption: there exists a notion of (relative) importance between pieces of information in a document (or utterance), regardless of whether we can detect this or not. The broad operations required for this task are briefly classified below

1.3.1 Preprocessing

First, a system takes a set of related texts as input and preprocesses them. The preprocessing step includes tokenization, Part-Of-Speech (POS) tagging, removal of stopwords and stemming. This preprocessing step allows us to obtain a more accurate representation of the information included in each sentence, and makes similarity measurement more efficient. Finally, select the best subset of sentences, based on the number of concepts each sentence holds, and finding the optimal combination of sentences that maximize informativity while minimizing redundancy.

1.3.2 Content Selection

This is the initial stage of a text summarizer. Here we choose what information from the input document(s) are relevant and should be used for generating the summary. While sentence compression and sentence fusion offer richer variations of sentences to include in a summary, one still needs to choose which sentences should be added. Optimally we must choose the sentences having the maximum informativeness while keeping a low redundancy. Content selection can be either supervised or unsupervised. In supervised content selection, domain specific parameters are fed to the system to train it to identify features relevant to the query provided by the user. In contrast, the unsupervised content selection generally incorporates statistical and mathematical tools such as TF-IDF scores, log-likelihood ratio, Jaccard scores or any other widely available similarity scores to identify the most informative features from the input document(s). As shown in the fig 1.1, content selection comprises many crucial sub stage such as sentence fragmentation, sentence extraction etc.

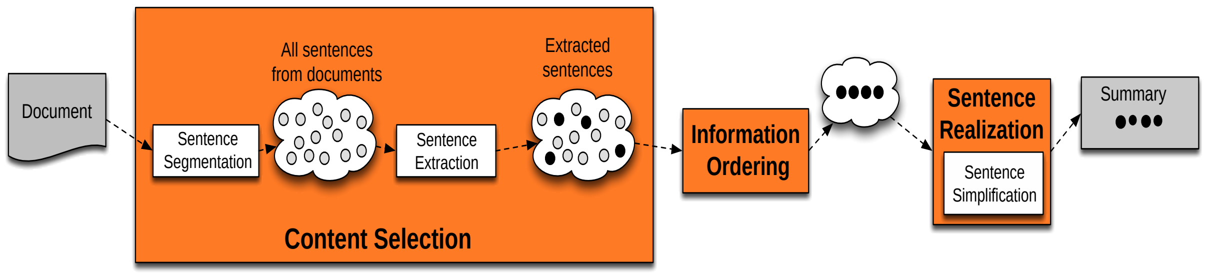

Figure 1.1: Generic Summarization Architecture

1.3.3 Sentence Clustering

The sentence clustering step allows us to regroup similar sentences in order to generate alternative sentences obtained by fusing sentences that belong to the same cluster. It correlates with the sentence ordering stage in fig 1.1. This is a crucial step as if we can't find enough clusters, we won't be able to generate any fused sentences, and if we are too broad during clustering we may try to fuse dissimilar sentences, thus resulting in incoherent fused sentences. To circumvent the risk of clustering together too many sentences, one can use Hierarchical Agglomerative Clustering with a complete-linkage strategy. This method proceeds incrementally, starting with each sentence considered as a cluster, and merging the two most similar clusters after each step. The complete-linkage strategy defines the similarity between two clusters as the lowest similarity score between two items of the clusters. Clusters may be small, but are highly coherent as each sentence they contain must be similar to every other sentence in the same cluster. A similarity threshold is set to stop the clustering process. If no cluster pair is found with a similarity above the threshold, the process stops, and the clusters are frozen. A similarity score of 0 is given when the two sentences do not have any words in common, as our sentence fusion module requires at least one word in common to operate.

1.3.4 Sentence Fusion

The most challenging aspect of the process is to meaningfully combine and prune the clustered output also maintaining syntactic and semantic integrity to form a grammatically coherent sentence structure. This step is not necessary in extractive summary as in extraction based summaries only extracted instances of sentences are placed together to form a summary. Sentence fusion is an intuitive process which can be implemented through various semantic, syntactic and statistical mechanisms. We discuss the various methods to do so in the upcoming sections. The fig 1.2 gives us a rudimentary idea about how sentence fusion takes place.

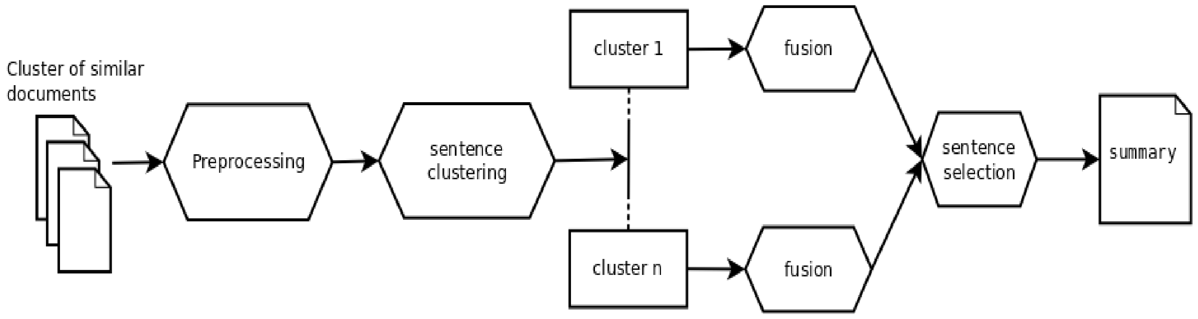

Figure 1.2: Sentence Fusion System Workflow

1.3.5 Sentence Realization

The final stage of summarization is generation. When the summary content has been created through abstracting and/or information extraction, the final output summary is generated using techniques of natural language generation, such as text planning, sentence planning, and sentence realization. Some of the common occurrences in this stage include removal of non-essential phrases, phrase merging, sentence fusion, sentence pruning etc.

2. Extractive Summarization

Content selection is the most important aspect of extraction based summaries. This is because the extracted contents are the final pieces with which the summary will be generated and unlike abstraction, there is no other room for improvement other than an optimal content selection. Numerous approaches for identifying important content for automatic text summarization have been developed to date. Topic representation approaches first derive an intermediate representation of the text that captures the topics discussed in the input. Based on these representations of topics, sentences in the input document are scored for importance. In contrast, in indicator representation approaches, the text is represented by a diverse set of possible indicators of importance which do not aim at discovering topicality. These indicators are combined, very often using machine learning techniques, to score the importance of each sentence. Finally, a summary is produced by selecting sentences in a greedy approach, choosing the sentences that will go in the summary one by one or globally optimizing the selection, choosing the best set of sentences to form a summary. In order to better understand the operation of summarization systems and to emphasize the design choices system developers need to make, we distinguish three relatively independent tasks performed by virtually all summarizers: creating an intermediate representation of the input which captures only the key aspects of the text, scoring sentences based on that representation and selecting a summary consisting of several sentences.

2.1 Intermediate Representation

Even the simplest systems derive some intermediate representation of the text they have to summarize and identify important content based on this representation. Topic representation approaches convert the text to an intermediate representation interpreted as the topic(s) discussed in the text. Some of the most popular summarization methods rely on topic representations and this class of approaches exhibits an impressive variation in sophistication and representation power. They include frequency, TF-IDF and topic word approaches in which the topic representation consists of a simple table of words and their corresponding weights, with more highly weighted words being more indicative of the topic; lexical chain approaches in which a thesaurus such as WordNet is used to find topics or concepts of semantically related words and then give weight to the concepts; latent semantic analysis in which patterns of word co-occurrence are identified and roughly construed as topics, as well as weights for each pattern; full blown Bayesian topic models in which the input is represented as a mixture of topics and each topic is given as a table of word probabilities (weights) for that topic. Indicator representation approaches represent each sentence in the input as a list of indicators of importance such as sentence length, location in the document, presence of certain phrases, etc. In graph models, such as LexRank (Radev and Gurkan, 2004), the entire document is represented as a network of inter-related sentences.

2.2 Score Sentences

Once an intermediate representation has been derived, each sentence is assigned a score which indicates its importance. For topic representation approaches, the score is commonly related to how well a sentence expresses some of the most important topics in the document or to what extent it combines information about different topics. For the majority of indicator representation methods, the weight of each sentence is determined by combining the evidence from the different indicators, most commonly by using machine learning techniques to discover indicator weights. In LexRank, the weight of each sentence is derived by applying stochastic techniques to the graph representation of the text.

2.3 Extractive Summarization Methods

1. Term Frequency-Inverse Document Frequency (TF-IDF) method
2. Cluster based method
3. Graph theoretic approach
4. Machine Learning approach
5. Latent Semantic Analysis
6. Neural Network approach
7. Query based extraction

2.3.1 TF-IDF Method

Bag-of-words model is built at sentence level, with the usual weighted term-frequency and inverse sentence frequency paradigm, where sentence-frequency is the number of sentences in the document that contain that term. These sentence vectors are then scored by similarity to the query and the highest scoring sentences are picked to be part of the summary. This is a direct adaptation of Information Retrieval paradigm to summarization. Summarization is query-specific, but can be adapted to be generic. It is a numerical statistic which reflects how important a word is in a given document. The TF-IDF value increases proportionally to the number of times a word appears in the document. This method mainly works in the weighted term-frequency and inverse sentence frequency paradigm .where sentence-frequency is the number of sentences in the document that contain that term. These sentence vectors are then scored by similarity to the query and the highest scoring sentences are picked to be part of the summary. Summarization is query-specific. The hypothesis assumed by this approach is that if there are ‘‘more specific words’’ in a given sentence, then the sentence is relatively more important. The target words are usually nouns. This method performs a comparison between the term frequency (tf) in a document -in this case each sentence is treated as a document and the document frequency (df), which means the number of times that the word occurs along all documents. The weighing exploits counts from a background corpus, which is a large collection of documents, normally from the same genre as the document that is to be summarized; the background corpus serves as indication of how often a word may be expected to appear in an arbitrary text. The only additional information besides the term frequency $c(w)$ that we need in order to compute the weight of a word w which appears $c(w)$ times in the input for summarization is the number of documents, $d(w)$, in a background corpus of D documents that contain the word. This allows us to compute the inverse document frequency:

$$\mathrm{TF} \times \mathrm{IDF} = \mathrm{c(w)}.\log\left(\frac{\mathrm{D}}{\mathrm{d(w)}}\right)$$

In many cases *c*(*w*) is divided by the maximum number of occurrences of any word in the document, which normalizes for document length. Descriptive topic words are those that appear often in a document, but are not very common in other documents. Words that appear in most documents will have an IDF close to zero. The TF*IDF weights of words are good indicators of importance, and they are easy and fast to compute.

2.3.2 Clustering Based Method

In multi-document summarization of news, the input by definition consists of several articles, possibly from different sources, on the same topic. Across the different articles there will be sentences that contain similar information. Information that occurs in many of the input documents is likely important and worth selecting in a summary. Of course, verbatim repetition on the sentence level is not that common across sources. Rather, similar sentences can be clustered together. In summarization, cosine similarity is the standard used to measure the similarity between the vector representations of sentences. In this approach, clusters of similar sentences are treated as proxies for topics; clusters with many sentences represent important topic themes in the input. Selecting one representative sentence from each main cluster is one way to produce an extractive summary, while minimizing possible redundancy in the summary. The sentence clustering approach to multi-document summarization exploits repetition at the sentence level. The more sentences there are in a cluster, the more important the information in the cluster is considered. Documents are usually written such that they address different topics one after the other in an organized manner. They are normally broken up explicitly or implicitly into sections. This organization applies even to summaries of documents. It is intuitive to think that summaries should address different "themes" appearing in the documents. Some summarizers incorporate this aspect through clustering. If the document collection for which summary is being produced is of totally different topics, document clustering becomes almost essential to generate a meaningful summary. Documents are represented using term frequency-inverse document frequency (TF-IDF) of scores of words. Term frequency used in this context is the average number of occurrences (per document) over the cluster. IDF value is computed based on the entire corpus. The summarizer takes already clustered documents as input. Each cluster is considered a theme. The theme is represented by words with top ranking term frequency, inverse document frequency (TF-IDF) scores in that cluster. Sentence selection is based on similarity of the sentences to the theme of the cluster C_i .The next factor that is considered for sentence selection is the location of the sentence in the document (L_i). In the context of newswire articles, the closer to the beginning a sentence appears, the higher its weight age for inclusion in summary. The last factor that increases the score of a sentence is its similarity to the first sentence in the document to which it belongs (F_i). The overall score (S_i) of a sentence i is a weighted sum of the above three factors:

$$S_i = W_1 \times C_i + W_2 \times F_i + W_3 \times L_i + \ldots.$$

Where S_i is the score of sentence C_i, F_i are the scores of the sentence i based on the similarity to theme of cluster and first sentence of the document it belongs to, respectively. Li is the score of the sentence based on its location in the document. W_1, W_2 and W_3 are the weights for linear combination of the three scores. The overall score (S_i) of a sentence i is a weighted sum

2.3.3 Graph Theoretic Approach

In this technique, there is a node for every sentence. Two sentences are connected with an edge if the two sentences share some common words, in other words, their similarity is above some threshold. This representation gives two results. The partitions contained in the graph (that is those sub-graphs that are unconnected to the other sub graphs), form distinct topics covered in the documents. The second result by the graph-theoretic method is the identification of the important sentences in the document. The nodes with high cardinality (number of edges connected to that node), are the important sentences in the partition, and hence carry higher preference to be included in the summary. Vertices represent sentences and edges between sentences are assigned weights equal to the similarity between the two sentences. The method most often used to compute similarity is cosine similarity with TF*IDF weights for words. Sometimes, instead of assigning weights to edges, the connections between vertices can be determined in a binary fashion: the vertices are connected only if the similarity between the two sentences exceeds a predefined threshold. Sentences that are related to many other sentences are likely to be central and would have high weight for selection in the summary. When the weights of the edges are normalized to form a probability distribution so that the weight of all outgoing edges from a given vertex sum up to one, the graph becomes a Markov chain and the edge weights correspond to the probability of transitioning from one state to another. Standard algorithms for stochastic processes can be used to compute the probability of being in each vertex of the graph at time *t* while making consecutive transitions from one vertex to next. As more and more transitions are made, the probability of each vertex converges, giving the stationary distribution of the chain. The stationary distribution gives the probability of (being at) a given vertex and can be computed using iterative approximation. Vertices with higher probabilities correspond to more important sentences that should be included in the summary. Graph-based approaches have been shown to work well for both single document and multi-document summarization. The graph theoretic method may also be adapted easily for visualization of inter- and intra-document similarity.

2.3.4 Machine Learning for Summarization

In this method, the training dataset is used for reference and the summarization process is modeled as a classification problem: sentences are classified as summary sentences and non-summary sentences based on the features that they possess. The classification probabilities are learnt statistically from the training data, using Bayes' rule

$$P(s\in < S | F1, F2, ..., FN) = P(F1, F2, ..., FN | s\in S) \times P(s\in S) / P(F1, F2, ..., FN)$$

Where s is a sentence from the document collection, F1, F2…FN are features used in classification. S is the summary to be generated and P (s∈ < S | F1, F2... FN) is the probability that sentence s will be chosen to form the summary given that it possesses features F1, F2…FN. In supervised methods for summarization, the task of selecting important sentences is represented as a binary classification problem, partitioning all sentences in the input into summary and non-summary sentences. A corpus with human annotations of sentences that should be included in the summary is used to train a statistical classifier for the distinction, with each sentences represented as a list of potential indicators of importance. The likelihood of a sentence to belong to the summary class, or the confidence of the classifier that the sentence should be in the summary, is the score of the sentence. The chosen classifier plays the role of a sentence scoring function, taking as an input the intermediate representation of the sentence and outputting the score of the sentence. The most highly scoring sentences are selected to form the summary, possibly after skipping some because of high similarity to already chosen sentences.

Machine learning approaches to summarization offer great freedom because the number of indicators of importance is practically endless. Any of the topic representation approaches discussed above can serve as the basis of indicators. Some common features include the position of the sentence in the document (first sentences of news are almost always informative), position in the paragraph (first and last sentences are often important), sentence length, similarity of the sentence with the document title or headings, weights of the words in a sentence determined by any topic representation approach, presence of named entities or cue phrases from a predetermined list, etc. Machine learning method has been applied for summarization. One important difference is whether the classifier assumes that the decision about inclusion in the summary is independently done for each sentence. This assumption is apparently not realistic, and methods that explicitly encode dependencies between sentences such as Hidden Markov Models and Conditional Random Fields outperform other learning methods. A problem inherent in the supervised learning paradigm is the necessity of labeled data on which classifiers can be trained. Asking annotators to select summary-worthy sentences is a reasonable solution but it is time consuming and even more importantly, annotator agreement is low and different people tend to choose different sentences when asked to construct an extractive summary of a text. Partly motivated by this issue and partly because of their interest in ultimately developing abstractive methods for summarization many researchers have instead worked with abstracts written by people (often professional writers). Researchers concentrated their efforts on developing methods for automatic alignment of the human abstracts and the input in order to provide labeled data of summary and non-summary sentences for machine learning. Some researchers have also proposed ways to leverage the information from manual evaluation of content selection in summarization in which multiple sentences can be marked as expressing the same fact that should be in the summary. Alternatively, one could compute similarity between sentences in human abstracts and those in the input in order to find very similar sentences, not necessarily doing full alignment. Another option for training a classifier is to employ a semi-supervised approach. In this paradigm, a small number of examples of summary and non-summary sentences are annotated by people. Then two classifiers are trained on that data, using different sets of features which are independent given the class or two different classification methods After that one of the classifiers is run on unannotated data, and its most confident predictions are added to the annotated examples to train the other classifier, repeating the process until some predefined halting condition is met. Several modifications to standard machine learning approaches are appropriate for summarization. In effect formulating summarization as a binary classification problem, which scores individual sentences, is not equivalent to finding the best summary, which consists of several sentences. In training a supervised model, the parameters may be optimized to lead to a summary that has the best score against a human model. For generic multi-document summarization of news, supervised methods have not been shown to outperform competitive unsupervised methods based on a single feature such as the presence of topic words and graph methods. Machine learning approaches have proved to be much more successful in single document or domain or genre specific summarization, where classifiers can be trained to identify specific types of information.

2.3.5 Latent Semantic Analysis

Singular Value Decomposition (SVD) is a very powerful mathematical tool that can find principal orthogonal dimensions of multidimensional data. It gets this name LSA because SVD applied to document word matrices, group documents that are semantically related to each other, even when they do not share common words. Words that usually occur in related contexts are also related in the same singular space. This method can be applied to extract the topic-words and content-sentences from documents. The advantage of using LSA vectors for summarization rather than the word vectors is that conceptual (or semantic) relations as represented in human brain are automatically captured in the LSA, while using word vectors without the LSA transformation requires design of explicit methods to derive conceptual relations. Since SVD finds principal and mutually orthogonal dimensions of the sentence vectors, picking out a representative sentence from each of the dimensions ensures relevance to the document, and orthogonality ensures non-redundancy. It is to be noted that this property applies only to data that has principal dimensions inherently—however, LSA would probably work since most of the text data has such principal dimensions owing to the variety of topics it addresses. Building the topic representation starts by filling in an n by m matrix A: each row corresponds to a word from the input (n words) and each column corresponds to a sentence in the input (m sentences). Entry a_{ij} of the matrix corresponds to the weight of word i in sentence j. If the sentence does not contain the word, the weight is zero, otherwise the weight is equal to the TF*IDF weight of the word. Standard techniques for singular value decomposition (SVD) from linear algebra are applied to the matrix A, to represent it as the product of three matrices: $A = U\Sigma V^T$ every matrix has a representation of this kind and many standard libraries provide a built-in implementation of the decomposition. Matrix U is a n by m matrix of real numbers. Each column can be interpreted as a topic, i.e. a specific combination of words from the input with the weight of each word in the topic given by the real number. Matrix Σ is diagonal m by m matrix. The single entry in row i of the matrix corresponds to the weight of the “topic”, which is the i^{th} column of U. Topics with low weight can be ignored, by deleting the last k rows of U, the last k rows and columns of Σ and the last k rows of V^T. This procedure is called dimensionality reduction. It corresponds to the thresholds employed in the centroid and topic words approaches, and topics with low weight are treated as noise. Matrix V^T is a new representation of the sentences, one sentence per row, each of which is expressed not in terms of words that occur in the sentence but rather in terms of the topics given in U. The matrix $D = \Sigma V^T$ combines the topic weights and the sentence representation to indicate to what extent the sentence conveys the topic, with d_{ij} indicating the weight for topic i in sentence j.

2.3.6 Neural Network Approach

This method involves training the neural networks to learn the types of sentences that should be included in the summary. This is accomplished by training the network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. This is done by a human reader. The neural network learns the patterns inherent in sentences that should be included in the summary and those that should not be included. It uses three-layered feed forward neural network, which has been proven to be a universal function approximator. The first phase of the process involves training the neural networks to learn the types of sentences that should be included in the summary. This is accomplished by training the network with sentences in several test paragraphs where each sentence is identified as to whether it should be included in the summary or not. This is done by a human reader. The neural network learns the patterns inherent in sentences that should be included in the summary and those that should not be included. Once the network has learned the features that must exist in summary sentences, we need to discover the trends and relationships among the features that are inherent in the majority of sentences. This is accomplished by the feature fusion phase, which consists of two steps:

1) Eliminating uncommon features

2) Collapsing the effects of common features.

The connections having very small weights after training can be pruned without affecting the performance of the network. As a result, any input or hidden layer neuron having no emanating connections can be safely removed from the network. In addition, any hidden layer neuron having no abutting connections can be removed. The hidden layer activation values for each hidden layer neuron are clustered utilizing an adaptive clustering technique. Each cluster is identified by its centroid and frequency. The activation value of each hidden layer neuron is replaced by the centroid of the cluster, which the activation value belongs to. This corresponds to collapsing the effects of common features. The combination of these two steps corresponds to generalizing the effects of features, as a whole, and providing control parameters for sentence ranking.

2.3.7 Query Based Extractive Method

In query based text summarization system, the sentences in a given document are scored based on the frequency counts of terms (words or phrases). The sentences containing the query phrases are given higher scores than the ones containing single query words. Then, the sentences with highest scores are incorporated into the output summary together with their structural context. Portions of text may be extracted from different sections or subsections. The resulting summary is the union of such extracts. The number of extracted sentences and the extent to which their context is displayed depends on the summary frame size which is fixed to the size of the screen that can be seen without scrolling. In the sentence extraction algorithm, whenever a sentence is selected for the inclusion in the summary, some of the headings in that context are also selected. The query based sentence extraction algorithm is as follows:

1: Rank all the sentences according to their score.

2: Add the main title of the document to the summary.

3: Add the first level-1 heading to the summary.

4: While (summary size limit not exceeded)

5: Add the next highest scored sentence.

6: Add the structural context of the sentence: (if any and not already included in the summary)

7: Add the highest level heading above the extracted text (call this heading h).

8: Add the heading before h in the same level.

9: Add the heading after h in the same level.

10: Repeat steps 7, 8 and 9 for the next highest level headings.

11: End while

An another query-specific summarization method views a document as a set of interconnected text fragments (passages) and focuses on keyword queries, since keyword search is the most popular information discovery method on documents, because of its power and ease of use. Firstly, at the preprocessing stage, it adds structure to every document, which can then be viewed as a labeled, weighted graph, called the document graph. Then, at query time, given a set of keywords, it performs keyword proximity search on the document graphs to discover how the keywords are associated in the document graphs. For each document its summary is the minimum spanning tree on the corresponding document graph that contains all the keywords. BAYESUM (Daume III and Marcu, 2006) is a model for sentence extraction in query-focused summarization. BAYESUM leverages the common case in which multiple documents are relevant to a single query. Using these documents as reinforcement for query terms, BAYESUM is not afflicted by the paucity of information in short queries. For a collection of D documents and Q queries, assume a D × Q binary matrix r, where r_{dq} = 1 iff document d is relevant to query q. In multi document summarization, r_{dq} will be 1 exactly when d is in the document set corresponding to query q.

2.4 Selecting summary Sentences

2.4.1 Greedy Approach

Greedy approach for both generic and query focused summarization that has been widely adopted is Maximal Marginal Relevance (MMR). In this approach, summaries are created using greedy, sentence by sentence selection. At each selection step, the greedy algorithm is constrained to select the sentence that is maximally relevant to the user query (or has highest importance score when a query is not available) and minimally redundant with sentences already included in the summary. MMR measures relevance and novelty separately and then uses a linear combination of the two to produce a single score for the importance of a sentence in a given stage of the selection process. To quantify both properties of a sentence, we can use cosine similarity. For relevance, similarity is measured to the query, while for novelty; similarity is measured against sentences selected so far. The MMR approach was originally proposed for query-focused summarization in the context of information retrieval, but could easily be adapted for generic summarization, for example by using the entire input as a user. This greedy approach of sequential sentence selection might not be that effective for optimal content selection of the entire summary. One typical problematic scenario for greedy sentence selection is when a very long and highly relevant sentence happens to be evaluated as the most informative early on. Such a sentence may contain several pieces of relevant information, alongside some not so relevant facts which could be considered noise. Including such a sentence in the summary will help maximize content relevance at the time of selection, but at the cost of limiting the amount of space in the summary remaining for other sentences. In such cases it is often more desirable to include several shorter sentences, which are individually less informative than the long one, but which taken together do not express any unnecessary information.

2.4.2 Global Summary Selection

Global optimization algorithms can be used to solve the new formulation of the summarization task, in which the best overall summary is selected. Given some constraints imposed on the summary, such as maximizing informativeness, minimizing repetition, and conforming to required summary length, the task would be to select the best summary. Finding an exact solution to this problem is NP-hard, but approximate solutions can be found using a dynamic programming algorithm. Exact solutions can be found quickly via search techniques when the sentence scoring function is local, computable only from the given sentence. Even in global optimization methods, informativeness is still defined and measured using features well-explored in the sentence selection literature. These include word frequency and position in the document, TF*IDF), similarity with the input, and concept frequency. Global optimization approaches to content selection have been shown to outperform greedy selection algorithms in several evaluations using news data as input, and have proved to be especially effective for extractive summarization of meetings. The performance of the approximate algorithm based on dynamic programming was lower, but comparable to that of the exact solutions. In terms of running time, the greedy algorithm is very efficient, almost constant in the size of the input. The approximate algorithm scales linearly with the size of the input and is thus indeed practical to use. The running time for the exact algorithm grows steeply with the size of the input and is unlikely to be useful in practice.

3. Summarization Using Sentence Fusion

Sentence fusion is one of the vital stages in abstractive summarization. We will discuss how different techniques exploit various sentence fusion methods to obtain their summaries. Our thesis work in particular relates to methods of sentence compression, sentence fusion, abstraction so it is necessary to understand how's and what's involved in these procedures. One should keep in mind that the work for the betterment of this process in ongoing and no one particular method is foolproof and the quest for more accurate results is still on. The methodologies describes below are ordered in a chronological fashion with respect to their inception so as to give an idea how this field developed itself in various aspects over the time.

3.1 Using Cross Document Structure Theory (CST)

One of the foremost ideas was the use of cross-document structure based on inter-document relationships such as paraphrase, citation, attribution, modality, and development. **Radev** argued that a CST-based analysis of related documents can facilitate multi-document summarization. Rhetorical Structure Theory (RST) (Mann & Thompson, 1988) is a comprehensive theory of text organization. It is based on "text coherence", or the presence in "carefully written text" of unity that would not appear in random sequences of sentences. RST posits the existence of relations among sentences. Most relations consist of one or more *nuclei* (the central components of a rhetorical relation) and zero or more *satellites* (the supporting components of the relation). An example of an RST relation is *evidence* which is decomposed into a nucleus (a claim) and a satellite (text that supports the claim). RST is intentionally limited to single documents. With CST, we attempt to describe the rhetorical structure of sets of related documents. Unlike RST, CST cannot rely on the deliberateness of writing style. However some observations of structure across documents which, while clearly not deliberate in the RST sense, can be quite predictable and useful. In a sense, CST associates certain behavior to a "collective document author" (that is, the collectivism of all authors of the related documents). A pioneering study in the typology of links among documents is described in (Trigg and Weiser, 1986). Trigg introduces taxonomy of link types across scientific papers. The 80 suggested link types such as *citation, refutation, revision, equivalence, and comparison* are grouped in two categories: *Normal* (inter-document links) and *Commentary* (deliberate cross-document links). While the taxonomy is quite exhaustive, it is by no means appropriate or intended for general domain texts (that is, other than scientific articles). A large deal of research in the automatic induction of document and hyper document structure is due to Salton's group at Cornell (Salton et al., 1997).SUMMONS (Radev and McKeown, 1998) is a knowledge-based multi-document summarization system, which produces summaries of a small number of news articles within the domain of terrorism. SUMMONS uses as input a set of semantic templates extracted by a message understanding system (Fisher et al., 96) and identify some patterns in them such as change of perspective, contradiction, refinement, agreement, and involved a large amount of knowledge engineering even for a relatively small domain of text and are not directly suitable for domain-independent text analysis.

3.1.1 Representing cross-document structure

Two complementary data structures to represent multi-document clusters are the *multi-document cube* and the *multi-document graph*

3.1.1.1 Multi-document Cube

A ***multi-document cube C*** is a three dimensional structure that represents related documents. The three dimensions are t (time), s (source) and p (position within the document).
A ***document unit U*** is a tuple (t,s,p). Document units can be defined at different levels of granularity, e.g., paragraphs, sentences, or words.
A ***document D*** is a sequence of document units $U_1U_2…U_n$ which corresponds to a one-dimensional projection of a multi-document cube along the source and time dimensions. Some additional concepts can be defined based on the above definitions.
A ***snapshot*** is a slice of the multi-document cube over a period of time Δt
An ***evolving document*** is a slice of the multi-document cube in which the source is fixed and time and position may vary.
An ***extractive summary*** *S* of a cube *C* is a set of document units, $S \subset C$,
A ***summarization operator*** transforms a cube *C* into a summary *S*

3.1.1.2 Multi-document Graphs

While multi-document cubes are a useful abstraction, they cannot easily represent text simultaneously at different levels of granularity (words, phrases, sentences, paragraphs, and documents). The second formalism that we introduce is the *multi-document graph*. Each graph consists of smaller sub-graphs for each individual document. Two types of links are used. The first type represents inheritance relationships among elements within a single document. These links are drawn using thicker lines. The second type represents semantic relationships among textual units.

3.1.2 Using CST for Information fusion

In this section it is described how CST can be used to generate personalized multi-document summaries from clusters of related articles in four steps: clustering, document structure analysis, link analysis, and personalized graph based summarization.
The first stage, clustering, can be either query independent (e.g., based on pure document similarity (Allan et al. 98)) or based on a user query (in which case clusters will be the sets of documents returned by a search engine).

The second stage, document analysis, includes the generation of document trees representing the sentential and phrasal structure of the document (Hearst 94, Kan et al. 98).

The third stage is the automatic creation and typing of links among textual spans across documents. Four techniques for identifying related textual units across documents can be used: *Lexical distance*
Lexical chains
Information extraction
Linguistic template matching

Lexical distance (Allan, 96) uses cosine similarity across pairs of sentences. Lexical chains (Barzilay & Elhadad, 97) are more robust than lexical matching as they take into account linguistic phenomena such as synonymy and hypernymy. The third technique, information extraction (Radev & McKeown, 98) identifies salient semantic roles in text (e.g., the place, perpetrator, or the effect of a terrorist event say for instance) and converts them to semantic templates. Two textual units are considered related whenever their semantic templates are related. Finally, a technique that will be used to identify some relationships such as citation, contradiction, and attribution is *template matching* which takes into account transformational grammar (e.g., relative clause insertion). For link *type analysis*, machine learning using lexical metrics and cue words is most appropriate.
The final step is summary extraction, based on the user-specified constraints on the summarizer. A graph-based operator defines a transformation on a multi-document graph (MDG) ***G*** which preserves some of its properties while reducing the number of nodes. An example of such an operator is the *link-preserving graph cover operator.* Its effect is to preserve only these nodes from the source MDG that are associated with the preferred cross-document links.

3.2 Cut and Paste Text Summarization

The name cut and paste might suggest an extractive based idea but this was one of earliest abstractive summarization technique developed by **Jing** (2000). This method was not pure abstraction as it incorporated many features of extraction based summarization. Back then, abstractive summarization had not developed to much extent and the primary focus of his work was to bridge the gap between automatically generated summaries and human-written abstracts. Instead of simple extraction of key features, this method aims to reuse the text in the given document to form summary. There are six independent modules present in this method which can form the summary from a given text and they work in tandem with syntactic knowledge, context and statistics learned from corpus analysis. It also decomposes human written abstracts to train and test the sentence reduction and sentence combination module. Around 300 human abstracts were analyzed for this technique.

3.2.1 Sentence Reduction

Extraneous phrases are removed. They can be at any granularity level a word, a phrase or a clause. The removal is done so that only spurious information is omitted keeping the relevant information intact. This is achieved by grammar checking, context information analysis and corpus analysis. The context information can be checked by extracting words and assigning importance score to them. Corpus of input articles and human generates abstracts are analyzed and three types of corpus probabilities are calculated which are probability that a phrase is completely removed, probability a phrase is partially removed and probability the phrase is unchanged. Taking into consideration all the steps we discussed so far final reduction takes place.

3.2.2 Sentence Combination

From analyzing the training set corpus, a set of rules are devised which are to be followed for sentence merging. The implementation of sentence combination involves joining two parse trees, substituting a subtree with one another or adding additional nodes. Formalism based on tree adjoining grammars was used for this action.

3.2.3 Syntactic Transformation

In both sentence reduction and combination, syntactic transformation may be involved such positional shift among words and phrases. Since this method is a supervised technique the manner in which such transformation is to be done depends upon the corpus and training dataset.

3.2.4 Lexical Paraphrasing

Common phrases are replaced with paraphrases. There must exist an external lexicon and dictionary from which such change can be made possible depending upon context.

3.2.5 Generalization

This stage re-arranges and replaces phrases and clauses with more general or specific description.

3.2.6 Reordering

The order of extracted sentences can be changed. For instance, like placing an ending sentence in an article at the beginning of an abstract.

3.3.6 System Architecture

Not only does this architecture given in Fig 3.1 pertain to the cut and paste summarization method but also provides a generic structure as to how most of the abstraction summarization process works. The input text is fed to the system. Sentence extraction techniques are incorporated to obtain key sentences of relevant information. The cut and paste generation module comes next which takes in the extracted sentences and performs sentence reduction and combination on them. This module takes help from the corpus of human abstracts which helps it to further analyze the sentences and make a better choice as to what should be removed and how the merging should be done. Also the parser, WordNet and various lexicons help with the paraphrasing task, re-arranging and reordering of the final text and final generation of the output summary. It must be kept in mind this method is quite primitive in its application and is highly dependent on user statistics but nevertheless it provides us with a rudimentary idea as to how one should proceed to the task of abstraction.

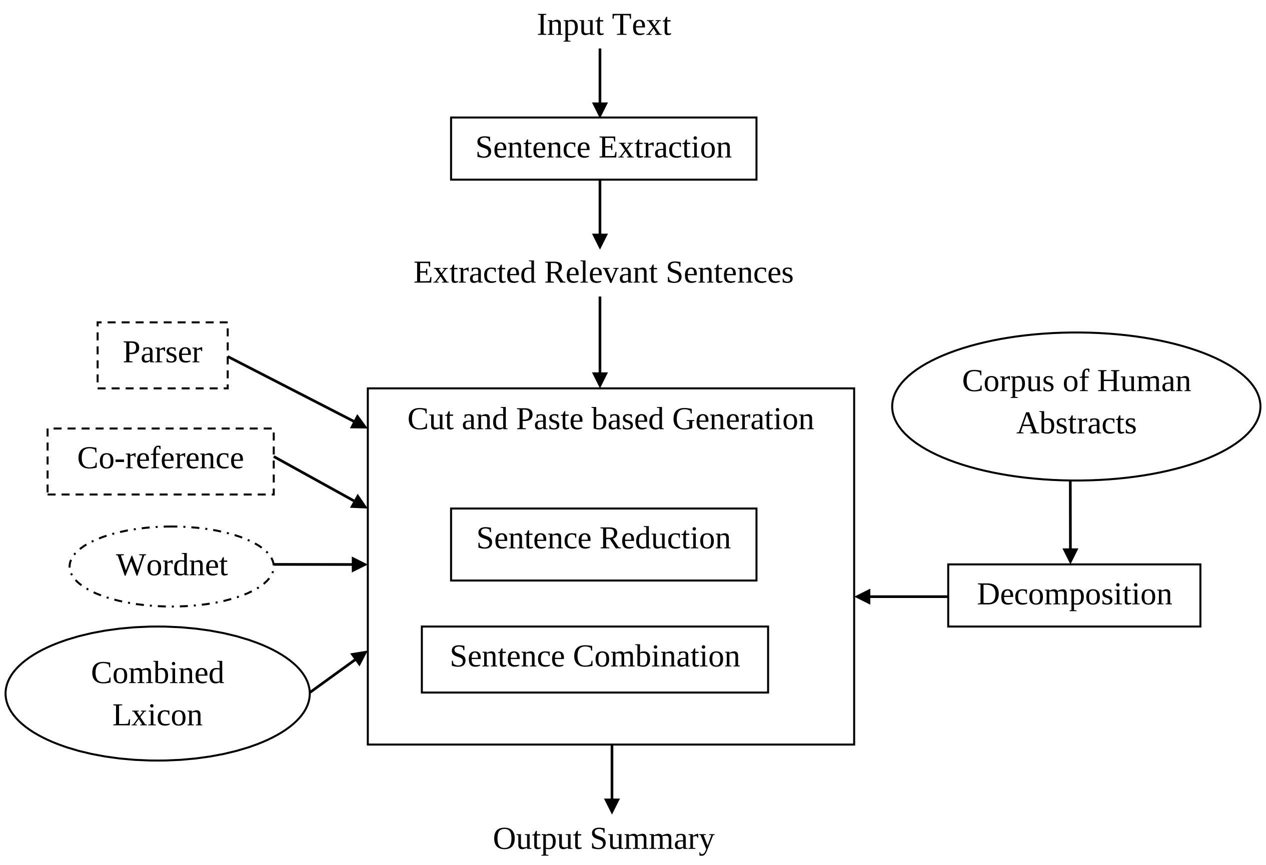

Figure 3.1: Cut and Paste Summarization Architecture

3.3 Text-to text Generation via Sentence Alignment

Text-to-text generation is an emerging area of research in NLP. Unlike in traditional concept-to-text generation, text-to-text generation applications take a text as input and transform it into a new text satisfying specific constraints, such as length in summarization or style in text simplification. One exciting new research direction is the automatic induction of such transformation rules. This is a particularly promising direction given that there are naturally occurring examples of comparable texts that convey the same information yet are written in different styles. Presented with two such texts, one can pair sentences that convey the same information, thereby building a training set of rewriting examples for the domain to which the texts belong. Automating this process will provide researchers in text-to-text generation with valuable training and testing resources. **Barzilay and Elhadad** investigate a novel approach informed by text structure for sentence alignment. This method emphasizes the search for an overall alignment, while relying on a simple local similarity function. They incorporate context into the search process in two complementary ways:

1. Mapping large text fragments using hypotheses learned in a supervised fashion.
2. Further refining the match through local alignment within mapping fragments to find sentence pairs.

When the documents in the collection belong to the same domain and genre, the fragment mapping takes advantage of the topical structure of the texts. This structure is derived in an unsupervised fashion by analyzing commonalities among texts on each side of the comparable corpora separately.
Given a comparable corpus consisting of two collections and a training set of manually aligned text pairs from the corpus, the algorithm follows four main steps. Steps 1 and 2 take place at training time. Steps 3 and 4 are carried out when a new text pair (Text1, Text2) is to be aligned.

1. **Topical structure induction**: by analyzing multiple instances of paragraphs within the texts of each collection, the topics characteristic of the collections are identified through clustering. Each paragraph in the training set gets assigned the topic it verbalizes.

2. Learning of structural mapping rules: using the training set, rules for mapping paragraphs are learned in a supervised fashion.

3. Macro alignment: given a new unseen pair (Text1, Text2), each paragraph is automatically assigned its topic. Paragraphs are mapped following the learned rules.

4. Micro alignment: for each mapped paragraph pair, a local alignment is computed at the sentence level. The final alignment for the text pair is the union of all the aligned sentence pairs

In the field of text generation, methods for representing the semantic structure of texts have been investigated through text schemata (McKeown, 1985) or rhetorical structures (Mann and Thompson, 1986). In this framework, the different topics of the text are identified, but much concern isn't shown with the relations holding between them or the order in which they typically appear. It is proposed to identify the topics typical to each collection in the comparable corpus by using clustering, such that each cluster represents a topic in the collection. The process of learning paragraph mapping rules is accomplished in two stages: first, identify the topics of each collection, Corpus1 and Corpus2, and label each paragraph with its specific topic. Second, using a training set of manually aligned text pairs, learn rules for mapping paragraphs from Corpus1 to Corpus2. Two paragraphs are considered mapped if they are likely to contain sentences that should be aligned.

3.3.1 Vertical Paragraph Clustering

A clustering at the paragraph level for each collection is performed. This stage is called Vertical Clustering because all the paragraphs of all the documents in Corpus1 get clustered, independently of Corpus2; the same goes for the paragraphs in Corpus2. At this stage, the only interesting thing is in identifying the topics of the texts in each collection, each cluster representing a topic. Then applying a hierarchical complete link clustering, similarity is calculated upon a simple cosine measure based on the word overlap of the paragraphs, ignoring function words. Since we want to group together paragraphs that convey the same type of information across the documents in the same collection, we replace all the text-specific attributes, such as proper names, dates and numbers, by generic tags. This way, it is ensured that two paragraphs are clustered not because they relate the same specific information, but rather, because they convey the same type of information.

3.3.2 Horizontal Paragraph Mapping

Once the different topics, or clusters, are identified inside each collection, this information can be used to learn rules for paragraph mapping (Horizontal Mapping between texts from Corpus1 and texts from Corpus2). Using a training set of text pairs, manually aligned at the sentence level, two paragraphs are considered to map each other if they contain at least one aligned sentence pair. The problem can be framed as a classification task: given training instances of paragraph pairs (P, Q) from a text pair, classify them as mapping or not. The features for the classification are the lexical similarity of P and Q, the cluster number of P, and the cluster number of Q. Here, similarity is again a simple cosine measure based on the word overlap of the two paragraphs. These features are weak indicators by themselves. Consequently, one can use the publicly available classification tool BoosTexter[1](Singer and Schapire, 2000) to combine them accurately.

3.3.3 Macro Alignment: Find Candidate Paragraph(s)

At this stage, the clustering and training are completed. Given a new unseen text pair (Text1, Text2), the goal is to find a sentence alignment between them. Two sentences with very high lexical similarity are likely to be aligned. We allow such pairs in the alignment independently of their context. This step allows catching the "easy" paraphrases. Focusing next on how the algorithm identifies the less obvious matching sentence pairs. For each paragraph in each text, identify the cluster in its collection it is the closest to. Similarity between the paragraph and each cluster is computed the same way as in the Vertical Clustering step, followed by apply mapping classification to find the mapping paragraphs in the text pair.

3.3.4 Micro Alignment: Find Sentence Pair(s)

Once the paragraph pairs are identified in (Text1, Text2), that we want to find, for each paragraph pair, the (possibly empty) subsets of sentence pairs which constitute a good alignment. Context is used in the following way: given two sentences with moderate similarity, their proximity to sentence pairs with high similarity can help us decide whether to align them or not. To combine the lexical similarity (again using cosine measure) and the proximity feature, we compute local alignments on each paragraph pair, using dynamic programming. The local alignment we construct fits the characteristics of the data that are considered. In particular, it is adapted to the framework to allow many-to-many alignments and some flips of order among aligned sentences. Given sentences i and j, their local similarity sim(i , j) is:

$$\text{sim}(i,j) = \cos(i, j) - \text{mismatch penalty}$$

The weight s(i,j) of the optimal alignment between the two sentences is computed as follows:

$$s(i,j) = \max \{ \begin{array}{l} s(i, j-1) - \text{skip penalty} \\ s(i-1, j) - \text{skip penalty} \end{array}$$

[1]https://www.cs.princeton.edu/~schapire/boostexter.html

$$\begin{aligned}&s(i-1, j-1) + sim(i,j)\\ &s(i-1, j-2) + sim(i,j) + sim(i, j-1)\\ &s(i-2, j-1) + sim(i,j) + sim(i-1, j)\\ &s(i-2, j-2) + sim(i, j-1) + sim(i-1, j) \}\end{aligned}$$

The mismatch penalty penalizes sentence pairs with very low similarity measure, while the skip penalty prevents *only* sentence pairs with high similarity from getting aligned.

3.4 Using Dependency Graph Compression

Filippova and Strube (2008) use groups of related sentences as input to a sentence fusion system and thus need to be identified first. Then the dependency trees of the sentences are modified and aligned. Syntactic importance and word informativeness scores are used to extract a new dependency tree from a graph of aligned trees. Finally, the tree is linearized.

3.4.1 Sentence Alignment

Sentence alignment for comparable corpora requires methods different from those used in machine translation for parallel corpora. For example, given two biographies of a person, one of them may follow the timeline from birth to death whereas the other may group events thematically or tell only about the scientific contribution of the person. Thus one cannot assume that the sentence order or the content is the same in two biographies. Shallow methods like word or bigram overlap, (weighted) cosine or Jaccard similarity are appealing as they are cheap and robust. In particular, (Nelken & Schieber, 2006) demonstrate the efficacy of a sentence-based *tf-idf* score when applied to comparable corpora. Following them, we define the similarity of two sentences *sim*(S1, S2) as

$$(S1 . S2) / (|S1|.|S2|) = \Sigma_t\, w_{s1}(t).w_{s2}(t) / (\Sigma_t w^2_{s1}(t).\, \Sigma_t w^2_{s2}(t))^{1/2}$$

Where S is the set of all lemmas but stop-words from s, and $w_S(t)$ is the weight of the term t:

$$w_S(t) = S(t)\, N^{-1}_t$$

Where S(t) is the indicator function of S, N_t is the number of sentences in the biographies of one person which contain t. Identical or nearly identical sentences (sim(s1, s2) > 0.8) are discarded and greedily build sentence clusters using a hierarchical group wise average technique is used. As a result, one sentence may belong to one cluster at most. These sentence clusters serve as input to the fusion algorithm.

3.4.2 Dependency Tree Modification

We apply a set of transformations to a dependency tree to emphasize its important properties and eliminate unimportant ones. These transformations are necessary for the compression stage.
Consider the sentence "Bohr studied mathematics and physics at the university in Copenhagen"

PREP preposition nodes (*an, in*) are removed and placed as labels on the edges to the respective nouns
CONJ a chain of conjuncts (*Mathematics and Physics*) is split and each node is attached to the parent node (*studied*) provided they are not verbs
APP a chain of words analyzed as appositions (*Niels Bohr*) is collapsed into one node
FUNC function words like determiners (*the*), auxiliary verbs or negative particles are removed from the tree and memorized with their lexical heads (memorizing negative particles preserves negation in the output)
ROOT every dependency tree gets an explicit root which is connected to every verb node
BIO all occurrences of the biographee (*Niels Bohr*) are replaced with the *bio* tag.

3.4.3 Node Alignment

Once group of two to four strongly related sentences and their transformed dependency trees are obtained, the aim is to find the best node alignment. Using a simple, fast and transparent method one can align any two words provided that they–

1. Are content words
2. Have the same part-of-speech
3. Have identical lemmas or are synonyms

In case of multiple possibilities, the choice is made randomly. By merging all aligned nodes a dependency graph is obtained which consists of all dependencies from the input trees. In case it contains a cycle, one of the alignments from the cycle is eliminated. This very simple method is preferred to the bottom-up ones (Barzilay & McKeown, 2005) for two main reasons. Pursuing local subtree alignments, bottom-up methods may leave identical words unaligned and thus prohibit fusion of complementary information. On the other hand, they may force alignment of two unrelated words if the subtrees they root are largely aligned. Although in some cases it helps discover paraphrases, it considerably increases chances of generating ungrammatical output which must be avoided at any cost.

3.4.4 Syntactic Importance Score

Given a dependency graph the objective is to get a new dependency tree from it. Intuitively, one can retain obligatory dependencies (e.g. *subject*) while removing less important ones (e.g. *adv*). When deciding on pruning an argument, previous approaches either used a set of hand-crafted rules (Barzilay and McKeown, 2005), or utilized a sub-categorization lexicon. The hand-crafted rules are often too general to ensure a grammatical argument structure for different verbs (e.g. *PPs can be pruned*). Sub-categorization lexicons are not readily available for many languages and cover only verbs. E.g. they do not tell that the noun *son* is very often modified by a PP using the preposition *of*, as in *the son of Niels Bohr,* and that the NP without a PP modifier may appear incomplete. To overcome these problems, they decide on pruning an edge by estimating the conditional probability of its label given its head, P(*l*|*h*).
For example, P(*subj*|*study*) – the probability of the label *subject* given the verb *study* – is higher than P(*in*|*study*), and therefore the subject will be preserved whereas the prepositional label and thus the whole PP can be pruned,

3.4.5 Word Informativeness Score

Retaining informative words in the output Tree are a necessity. There are many ways in which word importance can be defined. Here, they use a formula

$$I(w_i) = N^{-1}. f_i \log F_A / F_i$$

w_i is the topic word (either noun or verb), f_i is the frequency of w_i in the aligned biographies, F_i is the frequency of w_i in the corpus, and F_A is the sum of frequencies of all topic words in the corpus. *l* is the number of clause nodes above w and N is the maximum level of embedding of the sentence which w belongs to. By defining word importance differently, e.g. as relatedness of a word to the topic, one could apply this method to topic-based summarization

3.4.6 New Sentence Generation

The task of getting a tree from a dependency graph can be formulated as an optimization problem and solved ILP. In order to decide which edges of the graph to remove, for each directed dependency edge from head h to word w they introduce a binary variable $x^l_{h,w}$, where *l* stands for the label of the edge:

$x^l_{h,w} = 1$ if the dependency is preserved
0 otherwise

The goal is to find a subtree of the graph which gets the highest score of the objective function (1) to which both the probability of dependencies (P(*l*|*h*)) and the importance of dependent words (I(w)) contribute:

$$f(X) = \Sigma_x \; x^l_{h,w} \cdot P(l|h) \cdot I(w) \ldots\ldots\ldots\ldots(1)$$

The objective function is subject to three types of constraints presented below (W stands for the set of graph nodes minus root, i.e. the set of words).
STRUCTURAL constraints allow getting a tree from the graph: (2) ensures that each word has one head at most, (3) ensures connectivity in the tree, (4) is optional and restricts the size of the resulting tree to α words (α = min (0.6 · |W|, 10)).

$$\text{For all } w \,\varepsilon\, W,\ \Sigma_{h,l} \; x^l_{h,w} \leq 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$
$$\text{For all } w \,\varepsilon\, W,\ \Sigma_{h,l} \; x^l_{h,w} - |W|^{-1}\Sigma_{u,l} \; x^l_{w,u} \geq 0 \ldots\ldots\ldots\ldots(3)$$
$$\Sigma_x \; x^l_{h,w} \leq \alpha \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

SYNTACTIC constraints ensure the syntactic validity of the output tree and explicitly state which arguments should be preserved. We have only one syntactic constraint which guarantees that a subordinating conjunction (*sc*) is preserved if and only if the clause it belongs to serves as a subordinate clause (*sub*) in the output.

SEMANTIC constraints restrict coordination to semantically compatible elements. The idea behind these constraints is the following. It can be that one sentence says *He studied math* and another one *He studied physics*, so the output may unite the two words under coordination: *He studied math and physics*. But if the input sentences are *He studied physics* and *He studied sciences*, then one should not unite both, because *sciences* is the generalization of *physics*. Neither should one unite two unrelated words: *He studied with pleasure* and *He studied with Bohr* cannot be fused into *He studied with pleasure and Bohr*.
To formalize these intuitions we define two functions *hm(w,u)* and *rel(w,u)*: *hm(w,u)* is a binary function, whereas *rel(w,u)* returns a value from (0, 1). We also introduce additional variables $y^l_{w,u}$:

$$y^l_{w,u} = 1 \text{ if } h, l : x^l_{h,w} = 1 \;\&\&\; x^l_{h,u} = 1 \ldots\ldots\ldots\ldots\ldots\ldots (5)$$
$$0 \text{ otherwise}$$

For two edges sharing a head and having identical labels to be retained check in WordNet and in the taxonomy derived from Wikipedia (Kassner et al., 2008) that their dependents are not in the *h*yponymy or *m*eronymy relation (6). Verb coordination is prohibited unless it is found in one of the input sentences. If the dependents are nouns, also check that their semantic relatedness as measured with WikiRelate! (Strube & Ponzetto, 2006) is above a certain threshold (7). They empirically determined the value of ß = 0.36 by calculating an average similarity of coordinated nouns in the corpus.

$$\text{For all } y^l_{w,u}\ ,\ hm(w, u) \cdot y^l_{w,u} = 0 \ldots\ldots\ldots\ldots\ldots\ldots (6)$$
$$\text{For all } y^l_{w,u}\ ,\ (rel(w, u) - ß) \cdot y^l_{w,u} \geq 0 \ldots\ldots\ldots\ldots\ldots (7)$$

(6) Prohibits that *physics* (or *math*) and *sciences* appear together since, according to WordNet, *physics* is a hyponym of *science,* (7) blocks taking both *pleasure* and *Bohr* because *rel(pleasure,Bohr)* = 0.17. *math* and *physics* are neither in *IS-A*, nor *part-of* relation and are sufficiently related (*rel(Mathematics, Physics)* = 0.67) to become conjuncts.

3.4.7 Linearization

The “overgenerate-and-rank” approach to statistical surface realization is very common. Unfortunately, in its simplest and most popular version, it ignores syntactical constraints and may produce ungrammatical output. For example, an inviolable rule of German grammar states that the finite verb must be in the second position in the main clause. Since it is hard to enforce such rules with an n-gram language model, syntax-informed linearization methods have been developed for German. Applying this method to order constituents and, using the CMU toolkit, one can build a trigram language model from Wikipedia (approx. 1GB plain text) to find the best word order within constituents. Some constraints on word order are inferred from the input. Only inter-clause punctuation is generated.

3.5 Using Word Graphs

Filippova (2010) considers the task of summarizing a cluster of related sentences with a short sentence which is called *multi-sentence compression* and presents a simple approach based on shortest paths in word graphs. The advantage and the novelty of the proposed method is that it is syntax lean and requires little more than a tokenizer and a tagger. Given a cluster of similar, or related sentences, the most salient theme aim is summarizing in a short single sentence. Two challenges of sentence compression as well as text summarization are

(i) Important content selection
(ii) Its readable presentation.

Most existing systems use syntactic information to generate grammatical compressions. Incidentally, syntax also provides clues to what is likely to be important–e.g., the subject and the verb of the main clause are more likely to be important than a prepositional phrase or a verb from a relative clause. Of course, syntax is not the only way to gauge word or phrase importance. In the case of sentence compression being used for text summarization, one disposes of a rich context to identify important words or phrases. A well-known challenge for extractive multi-document summarization systems is to produce non-redundant summaries. There are two standard ways of avoiding redundancy: either one add sentences to the summary one-by-one and each time checks whether the sentence is significantly different from what is already there, or one clusters related sentences and selects only one from each cluster. In both cases a selected sentence may include irrelevant information, so one wishes to compress it, usually by taking syntactic and lexical factors into account. However, such an approach is suboptimal. Instead of compressing a single sentence, a *word graph* is built from all the words of the related sentences and compresses this graph. A word graph is a directed graph where an edge from word *A* to word *B* represents an *adjacency* relation. It also contains the *start* and *end* nodes. Word graphs have been widely used in natural language processing for building language models, paraphrasing, alignment, etc. Compared with dependency graphs, their use for sentence generation has been left largely unexplored, presumably because it seems that almost all the grammatical information is missing from this representation. Indeed, a link between a finite verb and an article does not correspond to any grammatical relation between the two. However, the premise for this work is that redundancy should be sufficient to identify not only important words but also salient links between words.

3.5.1 Word Graphs

Given a set of related sentences S = {s_1, s_2, ...s_n}, a word graph is built by iteratively adding sentences to it. As an illustration, consider the four sentences below and the graph obtained from them. Edge weights are omitted and italicized fragments from the sentences are replaced with dots for clarity.

(1) *The wife of a former U.S. president Bill* Clinton Hillary Clinton visited China last Mon-day.

(2) Hillary Clinton wanted to visit China last month *but postponed her plans* till Monday last week.

(3) Hillary Clinton paid *a visit to the People Re-public* of China on Monday.

(4) Last week the *Secretary of State* Ms. Clinton visited Chinese officials.

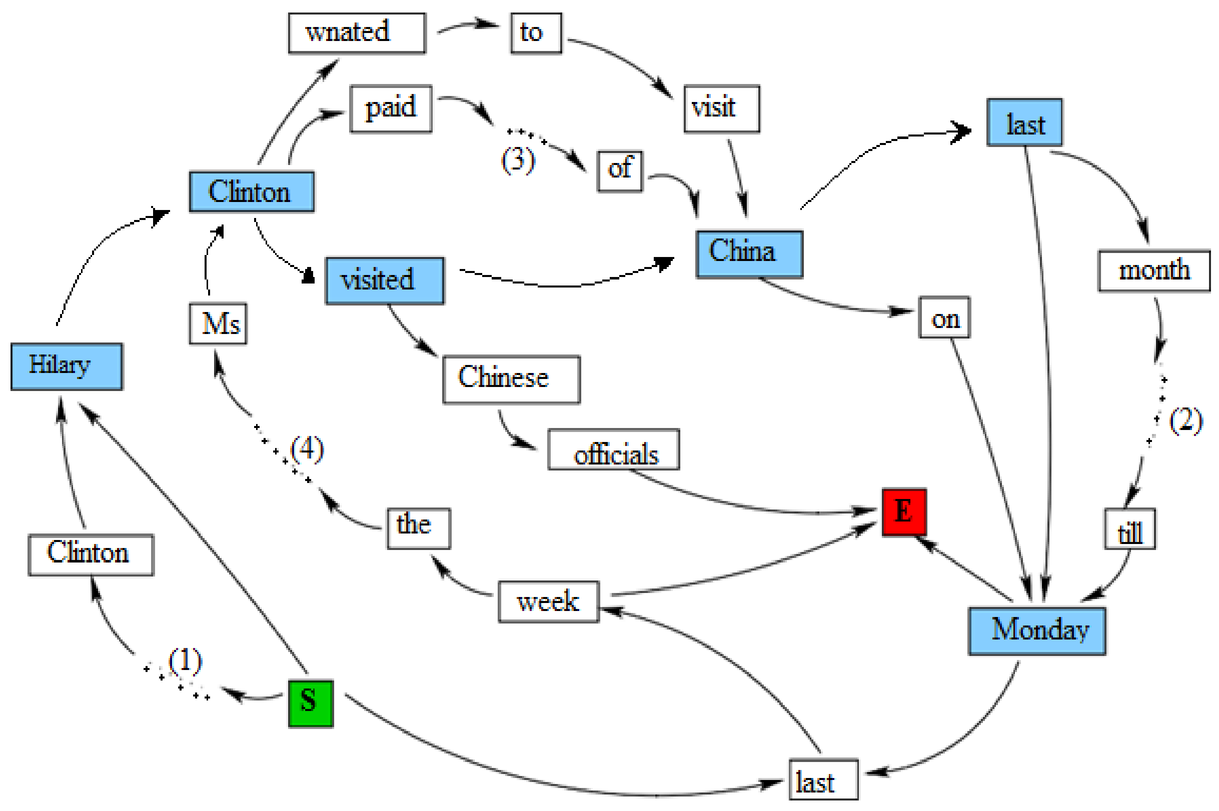

Figure 3.2: Filippova's Word Graph

A word from a sentence is mapped onto a node in the graph provided that they have the exact same lowercased word form and the same part of speech and that no word from this sentence has already been mapped onto this node. Using part of speech information reduces chances of merging verbs with nouns and generating ungrammatical sequences. If there is no candidate in the graph a new node is created. Word mapping and creation is done in three steps for the following three groups of words:

1. Non-stopwords for which no candidate exists in the graph or for which an unambiguous mapping is possible.

2. Non-stopwords for which there are either several possible candidates in the graph or which occur more than once in the sentence.

3. Stopwords.

This procedure is similar to the one used by (Barzilay and Lee, 2003) in that this also first identify "backbone nodes" (unambiguous alignments) and then add mappings for which several possibilities exist. However, they build lattices, i.e., directed acyclic graphs, whereas this graph may contain cycles. For the groups of words where mapping is ambiguous the immediate context (the preceding and following words in the sentence and the neighboring nodes in the graph) is checked and the candidate which has larger overlap in the context, or the one with a greater frequency (i.e., the one which has more words mapped onto it) is selected. For example, in word graph when sentence (4) is to be added, there are two candidate nodes for *last*. The one pointing to *week* is selected as *week* is the word following *last* in (4). Stopwords are mapped only if there is some overlap in non-stopword neighbors, otherwise a new node is created. Once all the words from the sentence are in place, we connect words adjacent in the sentence with directed edges. For newly created nodes, or nodes which were not connected before, we add an edge with a default weight of one. Edge weights between already connected nodes are increased by one. The same is done with the start and end nodes. Nodes store id's of the sentences their words come from as well as all their offset positions in those sentences. The described alignment method is fairly simple and guarantees the following properties of the word graph:

(i) Every input sentence corresponds to a loopless path in the graph

(ii) Words referring to the same entities or actions are likely to end up in one node

(iii) Stopwords are only joined in one node if there is an overlap in context.

The graph may generate a potentially endless amount of incomprehensible sequences connecting *start* and *end*. It is also likely to contain paths corresponding to good compressions, like the path connecting the nodes highlighted with blue in Figure 3.1. In the following we describe two methods of finding the best path, that is, the best compression for the input sentences.

3.5.2 Shortest Path as Compression

Characteristic of a good compression is it should neither be too long, nor too short. It should go through the nodes which represent important concepts but should not pass the same node several times. It should correspond to a likely word sequence. To satisfy these constraints we invert edge weights, i.e., link frequencies, and search for the shortest path (i.e., lightest in terms of the edge weights) from *start* to *end* of a predefined minimum length. This path is likely to mention salient words from the input and put together words found next to each other in many sentences. This is the first method. A minimum path length (in words) to eight is set which appears to be a reasonable threshold on development set–paths shorter than seven words were often incomplete sentences. Furthermore, to produce *informative* summaries which report about the main event of the sentence cluster, paths which do not contain a verb node are filtered.
For example, *Ozark's "Winter's Bone" at the 2010 Sundance Film Festival* might be a good title indicating what an article is about. However, it is not as informative as *"Winter's Bone" earned the grand jury prize at Sundance* which indeed conveys the gist of the event. Thus, *K* shortest paths are generated and all those which are shorter than eight words or do not contain a verb are filtered. The path with the minimum total weight is selected as the summary.

3.5.3 Improved Scoring and Re-ranking

The second configuration of a more sophisticated weighting function. The purpose of this function is two-fold:

(i) To generate a grammatical compression, it favors strong links, i.e., links between words which appear significantly often in this order

(ii) To generate an informative compression, it promotes paths passing through salient nodes.

3.5.3.1 Strong links: Intuitively, it is desirable that the compression path follow edges between words which are strongly associated with each other. Inverted edge frequency is not sufficient for that because it ignores the overall frequency of the nodes the edge connects. For example, edge frequency of three should count more if the edge connects two nodes with frequency of three rather than if their frequencies are much higher. Thus, we redefine edge weight as follows:

$$w(e_{i,j}) = [freq(i) + freq(j)] / freq(e_{i,j})$$

Furthermore, we also promote a connection between two nodes if there are multiple paths between them. For example, if some sentences speak of *president Barack Obama* or *president of the US Barack Obama*, and some sentences are about *president Obama*, we want to add some reward to the edge between *president* and *Obama*. However, longer paths between words are weak signals of word association. Therefore, the weight of an edge between the nodes i and j is reduced for every possible path between them but reduced proportionally to its length:

$$w'(e_{i,j}) = [freq(i) + freq(j)] / \Sigma s \in S\ diff(s, i, j)^{-1}$$

Where the function diff(s, i, j) refers to the distance between the offset positions (pos(s, i)) of words i and j in sentence s and is defined as follows:

$$diff(s, i, j) = pos(s, i) - pos(s, j), \text{ if } pos(s, i) < pos(s, j)$$
$$0 \text{ otherwise}$$

3.5.3.2 Salient words: The function above only indicates how strong the association between two words is. It assigns equal weights to edges connecting words encountered in a single sentence and words encountered next to each other in every sentence. To generate a summary concerning the most salient events and entities, the path is forced to go through most frequent nodes by decreasing edge weight with respect to the frequency of the nodes it connects. Thus, edge weight is further defined as follows:

$$w''(e_{i,j}) = w'(e_{i,j}) / (freq(i) \times freq(j))$$

The K-shortest paths algorithm is implemented to find the fifty shortest paths from *start* to *end* using the weighting function in. The paths which are shorter than eight words are filtered and which do not pass a verb node. Finally, re-rank the remaining paths by normalizing the total path weight over its length. This way the path which has the lightest average edge weight is obtained.

3.5.5 Baseline

As a first baseline we are searching for the most probable string with respect to the sentence cluster. In particular, the Viterbi algorithm is used to find the sequence of words of a predefined length n which maximizes the bigram probability (MLE based):

$$p(w_{1,n}) = p(w_1|s)p(w_2|w_1)\ldots p(e|w_n)$$

Similar to the shortest path implementation, compression length is specified and is set to eight tokens. However, the compressions obtained with this method are often unrelated to the main theme. The reason for that is that a token subsequence encountered in a single sentence is likely to get a high probability–all transition probabilities are equal to one–provided that the probability of entering this sequence is not too low. To amend this problem and to promote frequent words (i.e., words which are likely to be related to the main theme) maximize the following baseline score which takes into account both the bigram probabilities and the token likelihood, $p(w_i)$, which is also estimated from the sentence cluster:

$$b(w_{1,n}) = p(w_1|s)p(w_2|w_1)\ldots p(e|w_n)\pi_i\, p(w_i)$$

3.6 N-best Re-ranking in Multi Sentence Compression

Boudin and Morin (2013) extends the work of Filippova (2010) and slightly modifies her approach to output a slightly better model. In Filippova’s approach, punctuation marks are excluded. To generate well-punctuated compressions, they simply added a fourth step for adding punctuation marks in the graph. When mapping is ambiguous, they select the candidate which has the same immediate context. Once the words from a sentence are added to the graph, words adjacent in the sentence are connected with directed edges. Edge weights are calculated using the weighting function defined as:

$$w(i, j) = \text{cohesion}(i, j) / \text{freq}(i) \times \text{freq}(j)$$
$$\text{cohesion}(i, j) = \text{freq}(i) + \text{freq}(j) / \Sigma_{s \varepsilon S}\, d(s.i.j)^{-1}$$

Where freq(i) is the number of words mapped to the node i. The function d(s, i, j) refers to the distance between the offset positions of words i and j in sentence s. The purpose of this function is twofold:

(i) To generate a grammatical compression, links between words which appear often in this order are favored

(ii) To generate an informative compression, the weight of edges connecting salient nodes is decreased.

A K-shortest paths algorithm is then used to find the 50 shortest paths from start to end nodes in the graph. Paths shorter than eight words or that does not contain a verb are filtered. The remaining paths are re-ranked by normalizing the total path weight over its length. The path which has the lightest average edge weight is then considered as the best compression.

3.6.1 Re-ranking paths using Keyphrases

The main difficulty of MSC is to generate sentences that are both informative and grammatically correct. Here, redundancy within the set of input sentences is used to identify important words and salient links between words. Although this approach seemingly works well, important information is missing in 48% to 60% of the generated sentences (Filippova, 2010). One of the reasons for this is that node salience is estimated only with the frequency measure. To tackle this issue, they propose to re-rank the N-best list of compressions using keyphrases extracted from the set of related sentences. Intuitively, an informative sentence should contain the most relevant keyphrases. Re-ranking of generated compressions should be done according to the number and relevance of keyphrases they contain. Keyphrase extraction can be divided into two steps. First, a weighted graph is constructed from the set of related sentences, in which nodes represent words defined as word and POS tuples. Two nodes (words) are connected if their corresponding lexical units co-occur within a sentence. Edge weights are the number of times two words co-occur. TextRank (Mihalcea and Tarau, 2004), a graph-based ranking algorithm that takes into account edge weights, is applied for computing a salience score for each node. The score for node V_i is initialized with a default value and is computed in an iterative manner until convergence using this equation:

$$S(V_i) = (1-d)+d\times\Sigma_{Vj\ \varepsilon\ adj(Vi)}\ w_{ji}\ S(Vi)\ /\ \Sigma_{Vk\ \varepsilon\ adj(Vi)}\ w_{jk}$$

Where $adj(V_i)$ denotes the neighbors of V_i and d is the damping factor set to 0.85. The second step consists in generating and scoring keyphrase candidates. Sequences of adjacent words satisfying a specific syntactic pattern are collapsed into multi-word phrases. They use (ADJ)*(NPP|NC)+(ADJ)* for French, in which ADJ are adjectives, NPP are proper nouns and NC are common nouns. The score of a candidate keyphrase k is computed by summing the salience scores of the words it contains normalized by its length + 1 to favor longer n-grams

$$\text{score}(k) = \Sigma_{w\varepsilon k}\ \text{TextRank}(w)\ /\ (\text{length}(k) + 1)$$

The small vocabulary size as well as the high redundancy within the set of related sentences is two factors that make keyphrase extraction easier to achieve. On the other hand, a large number of the generated keyphrases are redundant. Some keyphrases may be contained within larger ones, e.g. giant tortoise and Pinta Island giant tortoise. To solve this problem, generated keyphrases are clustered using word overlap. For each cluster, select the keyphrase with the highest score. This filtering process enables the generation of a smaller subset of keyphrases while having a better coverage of the cluster content. Re-ranking techniques can suffer from the limited scope of the N-best list, which may rule out many potentially good candidates. For this reason, it’s advisable to use a larger number of paths than the one in (Filippova, 2010). Accordingly, the K-shortest paths algorithm is used to find the 200 shortest paths. Finally re-ranking the paths by normalizing the total path weight over its length multiplied by the sum of keyphrase scores it contains. The score of a sentence compression c is given by:

$$\text{score}(c) = \Sigma_{i,j\ \varepsilon\ \text{path}(c)}\ w(i,j)\ /\ \text{length}(c) \times \Sigma_{k\ \varepsilon\ c}\ \text{score}(k)$$

3.7 Using Single-Stage Inference

Previous approaches to fusion have often relied on dependency graph combination (Barzilay and McKeown, 2005), (Filippova and Strube, 2008) to produce an intermediate syntactic representation of the information in the sentence. Linearization of output fusions is usually performed by ranking hypotheses with a language model (LM), sometimes with language-specific heuristics to filter out ill-formed sentences. This approach is also known as overgenerate-and-rank and is often found to be a source of errors in T2T problems (Barzilay and McKeown, 2005). Although syntactic representations are natural for assembling text across sentences, recent work in unsupervised multi-sentence fusion has shown that well-formed output can often be constructed purely on the basis of adjacency relationships in a word graph (Filippova, 2010). Similarly, systems for related T2T tasks such as sentence compression and strict sentence intersection have also seen promising results by linearizing n-grams without explicitly relying on syntactic representations. **Thadani and McKeown** takes a similar perspective and assembles output text directly from n-grams over input tokens, but we employ a discriminative structured prediction approach in which likelihood under an LM is one of many features of output quality and parameters for all features are learned from a training corpus. Moreover, rather than rely on pipelined stages to first select the output content and then linearize an intermediate representation, this method jointly address token selection alongside phrase-based ordering thereby yielding a single stage approach to fusion.

3.7.1 ILP Formulation

The starting point for this work is the sequential structured transduction model of Thadani and McKeown (2013), originally devised for single sentence compression. This approach relies on integer linear programming (ILP) to find a globally optimal solution to generation problems involving heterogeneous substructures. ILP has been used frequently in recent T2T generation systems for sentence fusion and compression as well as other natural language processing tasks. Although LPs with integer constraints are NP-hard in the general case, the availability of optimized general-purpose ILP solvers and the natural limits on English sentence length make ILP inference attractive for sentence-level optimization problems. Consider a single fusion instance involving k source sentences $S= \{S_1 \ldots S_k\}$. The notation F_S is used to denote a fusion of the sentences in S. The inference step aims to retrieve the output sentence F^*_S that is the most likely fusion of S, i.e., the sentence that maximizes $p(F_s|S)$ or equivalently maximizes some scoring function score(F_S). In this feature-based discriminative setting, score(F_S) is defined as a dot product of weights w and a feature map $\mathbf{X}(S,F_S)$ defined over the fusion and its input; in other words

$$F^*_S = \text{argmax}_{F_S} w^T \mathbf{X}(S,F_S) \ldots\ldots\ldots\ldots(1)$$

The feature map **X** for an arbitrary fusion sentence is defined to factor over the words and potential n-grams from the input text. Let $T = \{t_i : 1 \leq i \leq N_j , 1 \leq j \leq |S|\}$ represent the set of tokens (including duplicates) in S and let x_i E{0,1} represent a token indicator variable whose value corresponds to whether token t_i is present in the output sentence F_S. Also consider n-gram phrases defined over the tokens in T and assume the use of bigrams without loss of generality. Let U represent the set of all possible bigrams that can be constructed from the tokens in T; in other words

$$U = \{<t_i, t_j> : t_i \varepsilon T \text{ u } \{START\}, t_j \varepsilon T \text{ u } \{END\}, i \text{ != } j\}$$

Following the notation for token indicators, let y_{ij} ε {0,1} represent a bigram indicator variable for whether the contiguous pair of tokens $<t_{i,}t_{j>}$ is in the output sentence. We represent entire token and bigram configurations with incidence vectors $x = <x_{i>t_i \varepsilon T}$ and $y= <y_{ij><t_i,t_j>\varepsilon U}$ which are equivalent to some subset of T and U respectively. With this notation, (1) can be rewritten as

$$F^*_S = \text{argmax}_{x,y} \; \Sigma_{t_i \varepsilon T} \; x_i \,.\, w^T_{tok} \, \Phi_{tok}(t_i) + \Sigma_{<t_i,t_j>\varepsilon U} \, y_{ij} \,.\, w^T_{ngr} \, \Phi_{ngr}(<t_i, t_j>)$$

$$= \text{argmax}_{x,y} \; x^T \Theta_{tok} + y^T \Theta_{ngr} \ldots\ldots\ldots\ldots\ldots(2)$$

Where Φ is a feature vector for tokens or bigrams and w is a corresponding vector of weight parameters. Each $\Theta = <w^T \Phi(s)>$ is therefore a vector of feature-based scores for either tokens or bigrams. The joint objective in (2) conveniently permits content-based features in Φ_{tok} for content selection and fluency features such as LM log-likelihoods in Φ_{ngr} for linearization. However, decoding a valid sentence with this objective is non-trivial. Merely selecting the tokens and bigrams that maximize (2) is liable to produce degenerate structures, i.e., cycles, disconnected components, branches and inconsistency between the token and bigram configurations in x and y. Most prior T2T linearization approaches such as the Viterbi-based approaches cannot be applied when the tokens in the input do not have a total ordering, as is the case when the input consists of more than one sentence.

3.7.2 Structural Constraints

Now briefly the structural constraints proposed by Thadani and McKeown (2013) are described to address the problem of degeneracy in sentential structure. First, one must consider the problem of output consistency—more formally, bigram variables y_{ij} that are non-zero must activate their token variables x_i and x_j while token variables can only activate a single bigram variable in the first and second position each.

$$x_i - \Sigma_j y_{ij} = 0 \text{ , for all } t_j \varepsilon T \text{ …………(3)}$$
$$xj - \Sigma_i y_{ij} = 0 \text{ , for all } t_i \varepsilon T \text{…………..(4)}$$

The second requirement for non-degenerate output is that non-zero y_{ij} must form a sentence like linear ordering of tokens, avoiding cycles and branching. For this purpose, auxiliary variables are introduced to establish single-commodity flow between all pairs of tokens that may appear adjacent in the output. Linear token ordering is maintained by defining real-valued commodity flow variables Γ_{ij} which are non-negative.

$$\Gamma_{ij} \geq 0, \text{ for all } <ti, tj> \varepsilon U \text{………….. (5)}$$

Each active token in the solution must have some positive incoming commodity and consumes one unit of this commodity, transmitting the remaining value to outgoing flow variables. This ensures that cycles cannot be present in the flow structure.

$$\Sigma_i \Gamma_{ij} - \Sigma_k \Gamma_{jk} = x_j \text{ , for all } t_j \varepsilon T$$

The acyclic flow structure can be imparted to y by constraining bigram indicators to be active only if their corresponding tokens have positive commodity flow between them.
$\Gamma_{ij} - C_{max} y_{ij} \leq 0$; for all $<t_{i,} t_j> \varepsilon$ U where C_{max} is the maximum amount of commodity that the Γ_{ij} variables may carry and serves as an upper bound on the number of output tokens. Finally, in order to establish connectivity in the output, they also introduce indicator variables y_{*j} and y_{i*} to denote the sentence-starting and terminating bigrams <START, t_j> and <$t_{i,}$ END> respectively. A valid output sentence must be started and terminated by exactly one bigram.

$$\Sigma_j y_{*j} = 1 \text{ and } \Sigma_i y_{i*} = 1$$

Flow variables y_{*j} and y_{i*} , are also defined for START and END respectively. Since START has no incoming flow variables, the amount of commodity in y_{*j} are unconstrained. This provides the only point of origin for the commodity and, in conjunction with (7), induces connectivity in y.

3.7.3 Redundancy constraints

While it is expected to get largely positive weights on features for supporting tokens, this will also have the effect of encouraging of more than one token from the same group to occur in the output. In order to avoid this problem, we add a constraint for each group $G_k \varepsilon G$ that prevents tokens within a group from appearing more than once.

$$\Sigma_{i:ti\ \varepsilon\ Gk}\ x_i \leq 1;\ \text{for all}\ G_k\ \varepsilon\ G$$

3.7.4 Features

The features Φ over tokens and bigrams that guide inference for fusion instances are described here.

1. Salience: Fluent output fusions might require specific words to be preserved, highlighted or perhaps rejected. This can be expressed through features on token variables that indicate a priori salience, for which one consider patterns of part-of-speech (POS) tags and dependency arc labels obtained from input parses. Specifically, they define indicator features for POS sequences of length up to 2 that surround the token and the POS tag of the token's syntactic governor conjoined with the label. Features for whether tokens appear within parentheses and if they are part of a capitalized sequence of tokens (an approximation of named entity markup) are also maintained.
2. Fluency: These features are intended to capture how the presence of a given bigram contributes to the overall fluency of a sentence. The bigram variables are scored with a feature expressing their log-likelihood under an LM. Also included are features that indicate the sequence of POS tags and dependency labels corresponding to the tokens an bigram variable covers
3. Fidelity: One might reasonably expect that many bigrams in the input sentences will appear unchanged in the output fusion. Therefore Boolean features are proposed that indicate whether a bigram was seen in the input.
4. Pseudo-normalization: A major drawback of using linear models for generation problems is an inability to employ output sentence length normalization when scoring structures. Word penalty features are used for this purpose following their use in machine translation (MT) systems. These features are simply set to 1 for every token and bigram and their parameters are intended to balance out biases in output length that are induced by other features.
5. Support: The amount of support— repetitions across input sentences—for nouns, verbs, adjectives and adverbs are noted. Features that count the number of repetitions for each of these tokens are defined, and conjoined with the POS class of each token. They also include binary variants of these features that indicate whether a token has support across 2, 3 or 4 input sentences. Each scale-dependent feature is recorded absolutely as well as normalized by the average length of an input sentence. This was done in order to encourage the model to be robust to variation in sentence length during training.

4. A Case Study: MultiGen

4.1 Overview

MultiGen[2] is part of the Columbia Summarization System[3]. It operates on a set of news articles describing the same event, creating a summary which synthesizes common information across documents. The system runs daily over real data within Newsblaster[4], a tool which collects news articles from multiple sources, organizes them into topical clusters and provides a summary for each of the clusters. In the case of multidocument summarization of articles about the same event, source articles can contain both repetitions and contradictions. Extracting all the similar sentences would produce a verbose and repetitive summary, while extracting only some of the similar sentences would produce a summary biased towards some sources. MultiGen uses a comparison of extracted similar sentences to select the appropriate phrases to include in the summary and reformulates them as new text. MultiGen consists of an analysis and a generation component. The analysis component (Hatzivassiloglou, Klavans, & Eskin, 1999) identifies units of text which convey similar information across the input documents using statistical techniques and shallow text analysis. Once similar text units are identified, they are clustered into themes. Themes are sets of sentences from different documents that contain repeated information and do not necessarily contain sentences from all the documents. For each theme, the generation component (Barzilay et al., 1999) identifies phrases which are in the intersection of the theme sentences and selects them as part of the summary. The intersection sentences are then ordered to produce a coherent text. At the end, for each theme there will be a single corresponding generated output sentence in the summary.

4.2 Theme Construction

The analysis component of MultiGen, Simfinder[5] (Hatzivassiloglou et al., 2001), identifies themes, groups of sentences from different documents that each says roughly the same thing. Each theme will ultimately correspond to at most one sentence in the output summary, generated by the fusion component, and there may be many themes for a set of articles. Sentences within a theme are not exact repetitions of each other; they usually include phrases expressing information that is *not* common to all sentences in the theme. If one of such sentences were used to represent the theme, the summary would contain extraneous information. Also, errors in clustering might result in the inclusion of some unrelated sentences. Evaluation involving human judges revealed that Simfinder identifies similar sentences with 49.3% precision at 52.9% recall. To identify themes, Simfinder extracts linguistically motivated features for each sentence, including WordNet synsets (Miller et al. 1990) and syntactic dependencies, such as subject–verb and verb–object relations. A log-linear regression model is used to combine the evidence from the various features into a single similarity value. The model was trained on a large set of sentences which were manually marked for similarity. The output of the model is a listing of real-valued similarity

[2]http://www.cs.columbia.edu/diglib/sumDemo/multiGen/main.html
[3]http://www.cs.columbia.edu/~hjing/summarization.html
[4]http://newsblaster.cs.columbia.edu
[5]http://academiccommons.columbia.edu/catalog/ac:160766

values on sentence pairs. These similarity values are fed into a clustering algorithm that partitions the sentences into closely related groups.

4.3 Theme Selection

To generate a summary of predetermined length, MultiGen induces a ranking on the themes and select the *n* highest. This ranking is based on three features of the theme: size measured as the number of sentences, similarity of sentences in a theme, and salience score. The first two of these scores are produced by Simfinder, and the salience score is computed using **lexical chains** as described below. Combining different rankings further filters common information in terms of salience. Since each of these scores has a different range of values, it performs ranking based on each score separately, then induce total ranking by summing ranks from individual categories:

Rank (theme) = *Rank* (Number of sentences in theme) + *Rank* (Similarity of sentences in theme) + *Rank* (Sum of lexical chain scores in theme)

Lexical chains—sequences of semantically related words—are tightly connected to the lexical cohesive structure of the text and have been shown to be useful for determining which sentences are important for single-document summarization (Barzilay and Elhadad, 1997). In the multi-document scenario, lexical chains can be adapted for theme ranking based on the salience of theme sentences within their original documents. Specifically, a theme that has many sentences ranked high by lexical chains as important for a single document summary is, in turn, given a higher salience score for the multi-document summary. In this implementation, a salience score for a theme is computed as the sum of lexical chain scores of each sentence in a theme.

4.4 Theme Ordering

Once the themes are filtered out that have a low rank, the next task is to order the selected themes into coherent text. The ordering strategy aims to capture chronological order of the main events and ensure coherence. To implement this strategy in MultiGen, it selects for each theme the sentence which has the earliest publication time (**theme time stamp**). To increase the coherence of the output text, it identifies blocks of topically related themes and then applies chronological ordering on blocks of themes using theme time stamps (Barzilay, Elhadad and McKeown, 2002). These stages produce a sorted set of themes which are passed as input to the sentence fusion component, described in the next section. In this section, two algorithms for ordering sentences are described suitable for multi-document summarization in the news genre. The first algorithm, Majority Ordering (MO), relies only on the original orders of sentences in the input documents. The second one, Chronological Ordering (CO), uses time-related features to order sentences.

4.4.1 Majority Ordering

In single document summarization, the order of sentences in the output summary is typically determined by their order in the input text. This strategy can be adapted to multi-document summarization. Consider two themes, Th_1 and Th_2; if sentences from Th_1 precede sentences from Th_2 in all input texts, then presenting Th_1 before Th_2 is likely to be an acceptable order. To use the majority ordering algorithm when the order between sentences from Th_1 and Th_2 varies from one text to another, one must augment the strategy. One way to define the order between Th_1 and Th_2 is to adopt the order occurring in the majority of the texts where Th_1 and Th_2 occur. This strategy defines a pairwise order between themes. However, this pairwise relation is not necessarily transitive. For example, given the themes Th_1, Th_2 and Th3 and the following situation: Th_1 precedes Th_2 in a text, Th_2 precedes Th_3 in the same text or in another text, and Th_3 precedes Th_1 in yet another text; there is a conflict between the orders (Th_1; Th_2; Th_3) and (Th_3; Th_1). Since transitivity is a necessary condition for a relation to be called an order, this relation does not form an order. Therefore, it has to expanded, this pairwise relation to provide a total order. In other words, we have to find a linear ordering between themes which maximizes the agreement between the orderings provided by the input texts. For each pair of themes, Th_i and Th_j , keep two counts, $C_{i,j}$ and $C_{j,i}$; $C_{i,j}$ is the number of input texts in which sentences from Th_i occur before sentences from Th_j, and $C_{j,i}$ is the same for the opposite order. The weight of a linear order ($Th_{i1}....Th_{ik}$) is defined as the sum of the counts for every pair $C_{il,im}$, such that $i_l \leq i_m$ and l,m ϵ {1…k}. Stating this problem in terms of a directed graph where nodes are themes, and a vertex from Th_i to Th_j has the weight $C_{i,j}$, the aim is to find a path with maximal weight which traverses each node exactly once, such a graph is called a precedence graph.

4.4.2 Chronological Ordering

Multi-document summarization of news typically deals with articles published on different dates, and articles themselves cover events occurring over a wide range of time. Using chronological order in the summary to describe the main events helps the user understand what has happened. It seems like a natural and appropriate strategy. To identify the date an event occurred requires a detailed interpretation of temporal references in articles. While there have been recent developments in disambiguating temporal expressions and event ordering, correlating events with the date on which they occurred is a hard task. In this case, the theme time is approximated by its first publication time i.e., the first time the theme has been reported in the set of input articles. It is an acceptable approximation for news events that the first publication time of an event usually corresponds to its occurrence in real life. For instance, in a terrorist attack story, the theme conveying the attack itself will have a date previous to the date of the theme describing a trial following the attack. Articles released by news agencies are marked with a publication time, consisting of a date and a time with two fields (hour and minutes). Articles from the same news agency are thus guaranteed to have different publication times. This is also quite likely for articles coming from different news agencies. During the development of MultiGen, hundreds of articles were processed, and never was such a situation encountered where two articles had the same publication time. Thus, the publication time serves as a unique identifier over articles. As a result, when two themes have the same publication time, it means that they both are reported for the first time in the same article. Chronological Ordering (CO) algorithm takes as input a set of themes and orders them chronologically whenever possible. Each theme is assigned a date corresponding to its first publication. To do so, select for each theme the sentence that has the earliest publication time. This is called the time stamp sentence and assigns its publication time as articles released by news agencies are marked with a publication time, consisting of a date and a time with two fields (hour and minutes).

4.5 Sentence Fusion

Given a group of similar sentences—a theme—the problem is to create a concise and fluent fusion of information, reflecting facts common to all sentences. To achieve this goal one needs to identify phrases common to most theme sentences, and then combine them into a new sentence. At one extreme, one might consider a shallow approach to the fusion problem, adapting the "bag of words" approach. However, sentence intersection in a set-theoretic sense produces poor results. The inadequacy of the bag-of-words method to the fusion task demonstrates the need for a more linguistically motivated approach. At the other extreme, previous approaches (Radev and McKeown, 1998) have demonstrated that this task is feasible when a detailed semantic representation of the input sentences is available. However, these approaches operate in a limited domain, where information extraction systems can be used to interpret the source text. The task of mapping input text into a semantic representation in a domain-independent setting extends well beyond the ability of current analysis methods. These considerations suggest that a new method is needed for the sentence fusion task. Ideally, such a method would not require a full semantic representation. Rather, it would rely on input texts and shallow linguistic knowledge (such as parse trees) that can be automatically derived from a corpus to generate a fusion sentence. In this approach, sentence fusion is modeled after the typical generation pipeline: content selection (what to say) and surface realization (how to say it). In contrast to that involved in traditional generation systems in which a content selection component chooses content from semantic units, our task is complicated by the lack of semantics in the textual input. At the same time, we can benefit from the textual information given in the input sentences for the tasks of syntactic realization, phrasing, and ordering, in many cases, constraints on text realization are already present in the input.

4.5.1 Identification of Common Information

The task is to identify information shared between sentences. This is done by aligning constituents in the syntactic parse trees for the input sentences. This alignment process differs considerably from alignment for other NL tasks, such as machine translation, because one cannot expect a complete alignment. Rather, a subset of the subtrees in one sentence will match different subsets of the subtrees in the others. Furthermore, order across trees is not preserved, there is no natural starting point for alignment, and there are no constraints on crosses. For these reasons a bottom-up local multi-sequence alignment algorithm is developed that uses words and phrases as anchors for matching. This algorithm operates on the dependency trees for pairs of input sentences, using a dependency-based representation because it abstracts over features irrelevant for comparison such as constituent ordering. Given a pair of sentences, determine which sentence constituents convey information appearing in both sentences. This algorithm will be applied to pairwise combinations of sentences in the input set of related sentences. The intuition behind the algorithm is to compare all constituents of one sentence to those of another and select the most similar ones. Of course, how this comparison is performed depends on the particular sentence representation used. A good sentence representation will emphasize sentence features that are relevant for comparison, such as dependencies between sentence constituents, while ignoring irrelevant features, such as constituent ordering. A representation which fits these requirements is a dependency-based representation (Melcuk, 1988).

4.5.1.1 Sentence Representation

The sentence representation is based on a **dependency tree,** which describes the sentence structure in terms of dependencies between words. The similarity of the dependency tree to a predicate–argument structure makes it a natural representation for a comparison. This representation can be constructed from the output of a traditional parser. In fact, they have developed a rule-based component that transforms the phrase structure output of Collins's parser into a representation in which a node has a direct link to its dependents. It also mark verb– subject and verb–node dependencies in the tree. The process of comparing trees can be further facilitated if the dependency tree is abstracted to a canonical form which eliminates features irrelevant to the comparison, hypothesizing that the difference in grammatical features such as auxiliaries, number, and tense has a secondary effect when the meaning of sentences is being compared. Therefore, they represent in the dependency tree only non-auxiliary words with their associated grammatical features. For nouns, it records their number, articles, and class (common or proper). For verbs, tense, mood (indicative, conditional, or infinitive), voice, polarity, aspect (simple or continuous), and taxis (perfect or none) are recorded. The eliminated auxiliary words can be re-created using these recorded features. The system also transforms all passive-voice sentences to the active voice, changing the order of affected children.

4.5.1.2 Alignment

The alignment of dependency trees is driven by two sources of information, the similarity between the structure of the dependency trees and the similarity between lexical items. In determining the structural similarity between two trees, take into account the types of edges (which indicate the relationships between nodes). An edge is labeled by the syntactic function of the two nodes it connects (e.g., subject– verb). It is unlikely that an edge connecting a subject and verb in one sentence, for example, corresponds to an edge connecting a verb and an adjective in another sentence. The word similarity measures take into account more than word identity. They also identify pairs of paraphrases, using WordNet and a paraphrasing dictionary. It automatically constructs the paraphrasing dictionary from a large comparable news corpus using the co-training method. The dictionary contains pairs of word-level paraphrases as well as phrase-level paraphrases. During alignment, each pair of non-identical words that do not comprise a synset in WordNet is looked up in the paraphrasing dictionary; in the case of a match, the pair is considered to be a paraphrase.
Now an intuitive explanation of how their tree similarity function, denoted by *Sim*, is computed. If the optimal alignment of two trees is known, then the value of the similarity function is the sum of the similarity scores of aligned nodes and aligned edges. Since the best alignment of given trees is not known a priori, the maximal score among plausible alignments of the trees is selected. Instead of exhaustively traversing the space of all possible alignments, recursively construct the best alignment for trees of given depths, assuming that we know

how to find an optimal alignment for trees of shorter depths. More specifically, at each point of the traversal two cases must be considered. In the first case, two top nodes are aligned with each other and their children are aligned in an optimal way by applying the algorithm to shorter trees. In the second case, one tree is aligned with one of the children of the top node of the other tree; again we can apply our algorithm for this computation, since we decrease the height of one of the trees.
When T is a tree with root node v, let $c(T)$ denote the set containing all children of v.
For a tree T containing a node s, the subtree of T which has s as its root node is denoted by Ts.
Given two trees T and $T^*$with root nodes v and v^*, respectively, the similarity $Sim(T, T^*)$ between the trees is defined to be the maximum of the three expressions $NodeCompare(T,T^*)$, $\max_{s\in c(T)} Sim(Ts, T^*)$, and $\max_{s^*\in c(T^*)} Sim(T, T^*_{s^*})$.

The maximization in the *NodeCompare* formula searches for the best possible alignment for the child nodes of the given pair of nodes and is defined by

$$NodeCompare(T, T^*) = NodeSimilarity(v, v^*)$$

$$+ \max_{m\in M(c(T),c(T^*))}(\Sigma_{(s,s^*)\in m}(EdgeSimilarity((v,s),(v^*,s^*)) + Sim(T_s, T^*_{s^*})))$$

where $M(A,A^*)$ is the set of all possible matchings between A and A^*, and a matching (between A and A^*) is a subset m of $A \times A^*$such that for any two distinct elements (a, a^*), $(b, b^*) \in m$, both $a != b$ and $a^* != b^*$. In the base case, when one of the trees has depth one, $NodeCompare(T, T^*)$ is defined to be $NodeSimilarity(v, v^*)$.

The similarity score $NodeSimilarity(v, v^*)$ of atomic nodes depends on whether the corresponding words are identical, paraphrases, or unrelated. The similarity scores for pairs of identical words, pairs of synonyms, pairs of paraphrases, and edges are manually derived using a small development corpus. While learning of the similarity scores automatically is an appealing alternative, its application in the fusion context is challenging because of the absence of a large training corpus and the lack of an automatic evaluation function. The similarity of nodes containing flattened subtrees, such as noun phrases, is computed as the score of their intersection normalized by the length of the longest phrase. The similarity function *Sim* is computed using bottom-up dynamic programming, in which the shortest subtrees are processed first. The alignment algorithm returns the similarity score of the trees as well as the optimal mapping between the subtrees of input trees. In the resulting tree mapping, the pairs of nodes whose *NodeSimilarity* positively contributed to the alignment are considered parallel. Every node in one tree is mapped to at most one node in another tree. This restriction is necessary because the problem of optimizing many-to-many alignments is NP-hard. The subtree flattening performed during the preprocessing stage aims to minimize the negative effect of the restriction on alignment granularity. Another important property of this algorithm is that it produces a local alignment. Local alignment maps local regions with high similarity to each other rather than creating an overall optimal global alignment of the entire tree. This strategy is more meaningful when only partial meaning overlap is expected between input sentences, as in typical sentence fusion input. Only these high-similarity regions, which are called **intersection subtrees,** are included in the fusion sentence.

4.5.2 Fusion Lattice Computation

Fusion lattice computation is concerned with combining intersection subtrees. During this process, the system will remove phrases from a selected sentence, add phrases from other sentences, and replace words with the paraphrases that annotate each node. Among the many possible combinations of subtrees, we are interested only in those combinations which yield semantically sound sentences and do not distort the information presented in the input sentences. Exploring every possible combination is infeasible, since the lack of semantic information in the trees prohibits us from assessing the quality of the resulting sentences. Instead, a combination already present in the input sentences is selected as a basis and transformed into a fusion sentence by removing extraneous information and augmenting the fusion sentence with information from other sentences. The advantage of this strategy is that, when the initial sentence is semantically correct and the applied transformations aim to preserve semantic correctness, the resulting sentence is a semantically correct one. The three steps of the fusion lattice computation are as follows: selection of the **basis tree,** augmentation of the tree with alternative verbalizations, and pruning of the extraneous subtrees. Alignment is essential for all the steps. The selection of the basis tree is guided by the number of intersection subtrees it includes; in the best case, it contains all such subtrees. The basis tree is the centroid of the input sentences— the sentence which is the most similar to the other sentences in the input. Using the alignment-based similarity score described earlier, one identify the centroid by computing for each sentence the average similarity score between the sentence and the rest of the input sentences, then selecting the sentence with the highest score. Next, augment the basis tree with information present in the other input sentences. More specifically, by adding alternative verbalizations for the nodes in the basis tree and the intersection subtrees which are not part of the basis tree. The alternative verbalizations are readily available from the pairwise alignments of the basis tree with other trees in the input computed in the previous section. For each node of the basis tree, record all verbalizations from the nodes of the other input trees aligned with a given node. A verbalization can be a single word, or it can be a phrase, if a node represents a noun compound or a verb with a particle. Finally, subtrees which are not part of the intersection are pruned off the basis tree. However, removing all such subtrees may result in an ungrammatical or semantically flawed sentence; for example, we might create a sentence without a subject. This over-pruning may happen if either the input to the fusion algorithm is noisy or the alignment has failed to recognize similar subtrees. Therefore, a more conservative pruning is performed, deleting only the self-contained components which can be removed without leaving ungrammatical sentences. Such components include a clause in the clause conjunction, relative clauses, and some elements within a clause (such as adverbs and prepositions). Once these subtrees are removed, the fusion lattice construction
is completed.

4.6 Generation

The final stage in sentence fusion is linearization of the fusion lattice. Sentence generation includes selection of a tree traversal order, lexical choice among available alternatives, and placement of auxiliaries, such as determiners. Their generation method utilizes information given in the input sentences to restrict the search space and then chooses among remaining alternatives using a language model derived from a large text collection. For the word-ordering task, one does not have to consider all the possible traversals, since the number of valid traversals is limited by ordering constraints encoded in the fusion lattice. However, the basis lattice does not uniquely determine the ordering. The placement of trees inserted in the basis lattice from other theme sentences is not restricted by the original basis tree. While the ordering of many sentence constituents is determined by their syntactic roles, some constituents, such as time, location and manner circumstantial, are free to move. Therefore, the algorithm still has to select an appropriate order from among different orders of the inserted trees. The process so far produces a sentence that can be quite different from the extracted sentence; although the basis sentences provides guidance for the generation process, constituents may be removed, added in, or reordered. Wording can also be modified during this process. Although the selection of words and phrases which appear in the basis tree is a safe choice, enriching the fusion sentence with alternative verbalizations has several benefits. In applications such as summarization, in which the length of the produced sentence is a factor, a shorter alternative is desirable. This goal can be achieved by selecting the shortest paraphrase among available alternatives. Alternate verbalizations can also be used to replace anaphoric expressions, for instance, when the basis tree contains a noun phrase with anaphoric expressions (e.g., *his visit*) and one of the other verbalizations is anaphora-free. Substitution of the latter for the anaphoric expression may increase the clarity of the produced sentence, since frequently the antecedent of the anaphoric expression is not present in a summary. Moreover, in some cases substitution is mandatory. As a result of subtree insertions and deletions, the words used in the basis tree may not be a good choice after the transformations, and the best verbalization might be achieved by using a paraphrase of them from another theme sentence. The task of auxiliary placement is alleviated by the presence of features stored in the input nodes. In most cases, aligned words stored in the same node have the same feature values, which uniquely determine an auxiliary selection and conjugation. However, in some cases, aligned words have different grammatical features, in which case the linearization algorithm needs to select among available Linearization of the fusion sentence involves the selection of the best phrasing and placement of auxiliaries as well as the determination of optimal ordering. Since MultiGen system do not have sufficient semantic information to perform such selection, their algorithm is driven by corpus-derived knowledge. It generates all possible sentences from the valid traversals of the fusion lattice and score their likelihood according to statistics derived from a corpus. This approach is a standard method used in statistical generation. They trained a trigram model with Good–Turing smoothing over 60 megabytes of news articles collected by Newsblaster using the second version CMU–Cambridge Statistical Language Modeling toolkit. The sentence with the lowest length-normalized entropy (the best score) is selected as the verbalization of the fusion lattice.

4.7 Evaluation

Sentence fusion and summarization system are notoriously hard to evaluate against previous systems given that there is no standard domain or unified corpora to produce a gold standard baseline. Of the existing systems, all have their respective merits and flaws. Barzilay & McKeown's system can be taken as a non-trivial baseline since no other system in general can outperform theirs. It is also difficult to evaluate generation and summarization systems as there are many dimensions in which the quality of the output can be assessed. In general, two evaluation methods are followed; one that uses automatic ROGUE measure and one that manually evaluates the generated sentence.

The ROUGE (Lin, 2004) measure is based on n-gram recall between the generated summary and the human-written gold abstracts. ROUGE-2 for instance, corresponds to the following formula:

$$\text{ROGUE-2} = \Sigma_R \, \Sigma_{bi \, \varepsilon \, R} \, \text{Count}_{match} \, (b_i) \, / \, \Sigma_R \, \Sigma_{bi \, \varepsilon \, R} \, \text{Count} \, (b_i)$$

Where R is the set of reference summaries, b_i ε R are the bigrams in the current reference summary, Count_{match} (b_i) is the number of bigrams that are both in the candidate summary and current reference summary and Count (b_i) is the number of bigrams in the current reference summary.
As for manual evaluation, other than (Barzilay and McKeown, 2005), (Filippova, 2010) and (Boudin and Morin, 2013) are used to evaluate the grammaticality of the fused sentences on a 3-points scale: perfect (2 pts), if the fusion is a complete grammatical sentence; almost (1 pt) if it requires minor editing, e.g. one mistake in articles, agreement or punctuation; ungrammatical (0 pts), if it is none of the above.

5. Experiment

Text summarization can be effectively classified into two broad categories namely abstractive and extractive summarization. So far we discussed the previous works that have existed in the sentence fusion techniques and as we mentioned earlier our system is a compression based summarization system. Our work attempts to reach a compromise between these two extremes and formulate a summary containing elements from both the kinds. For doing so, we used two distinct methods and combined them. First we implemented a graph based technique to achieve sentence compression and information fusion. In the second step, we put grammatical rule based syntactic and semantic constraint to finally obtain a coherent and meaningful set of sentences for the complete summary. The system generated summaries are flexible in the sense that it does not have a pre-defined output length; it can generate output for any user-defined compression rate. We evaluated our system generated summaries by comparing them against the Opinosis gold summaries and our results on the Opinosis dataset are comparable with the Opinosis system and significantly better than the MEAD system.

5.2 Sentence Compression

Sentence compression can be defined as the method for obtaining shorter and more precise sentences from a group of similar sentences while maintaining a syntactic and semantic structure such that the result is grammatically coherent and informatively non-redundant. The sentence compression strategy used in (Filippova, 2010) requires only a POS tagger and list of stopwords to work. The NLTK[6] suit was used for the POS tagging as well as for obtaining the stopwords. The word graph which represents the cluster of similar sentences operates on the assumption that redundancy in a given set of similar sentences is enough to generate informative texts comprising the important terms because presence of redundancy will ensure that spurious words which do not have much association with other words will get filtered out since the weighting function is designed in such a way that redundant nodes are assigned less importance.

5.2.1 Word Graph Construction

Given a set of similar sentences, a word graph is a weighted directed graph where the nodes represent the words of a sentence while the edges represent the connectivity between two adjacent words. At the beginning the word graph has only a start and an end node. The steps for a word graph generation are as follows.

a. Starting with the first sentence, the words are added as nodes to the graph one by one.
b. With the addition of every new node, a directed edge from its previous node to the new node is created.
c. With every new node, the connecting directed edges acquire an edge weight of 1.

[6] http://www.nltk.org

d. If such a word is encountered which has its equivalent node with a same lowercased form and POS tag then no new node is generated rather the word is mapped onto that existing node.

e. Edge weights are incremented by 1 for the inclusion of an equivalent word node.

The inclusion of words in the graph is carried out in the following order.

a. Non-stopwords for which no candidates exist in the graph or for which an ambiguous mapping is possible.
b. Non-stopwords for which there are either several possible candidates in the graph or which occur more than once in a sentence.
c. Stopwords.

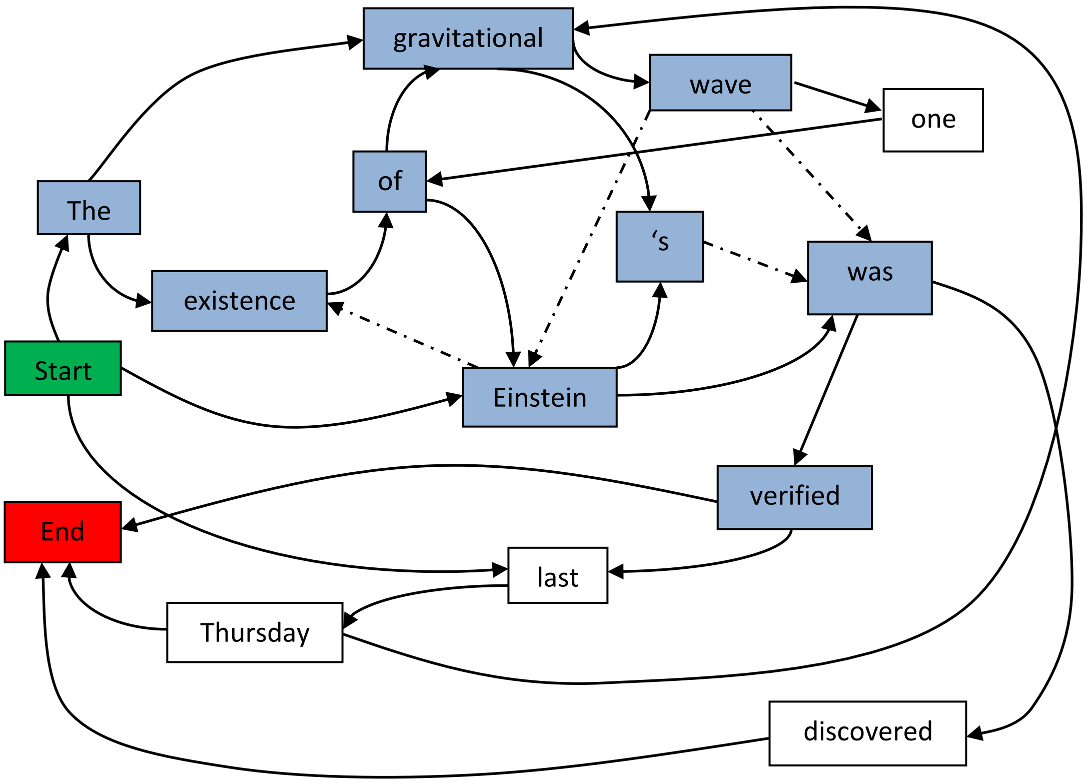

Figure 5.1: Word Graph generated from sentences (1-4) and a possible compression path

5.2.2 Illustration of Word Graph

Let us consider the 4 sentences (1−4) to help understand the workings of a word graph. Edge weights are omitted and the italicized fragments in the sentences are replaced by dotted lines in the figure for the ease of understanding.

The word graph generated from the 4 sentences is shown in Fig 5.1.

1. The gravitational wave, one of Einstein's *predictions* was verified last Thursday
2. Einstein *predicted the* existence of gravitational wave *and it* was verified

3. Last Thursday, gravitational wave was discovered
4. The existence of gravitational wave *predicted by* Einstein was verified

In case of ambiguous word mapping i.e. in case there exists such words for which multiple nodes during mapping is possible, the context in which the word is present, its neighboring nodes and the edge frequencies on the node in question are checked before assigning that word to any node.
Once the graph is generated and every edge has their respective edge weights labeled on them, the initial edge weight (discussed in the next section) is applied for every edge between every edge pair to calculate the resultant compression path.

5.2.3 Compressed Path

While calculating for the compressed path, we set a minimum threshold of 8 words for the formation of a sentence and the presence of a verb node. We choose 8 words as any less might lead to incomplete sentences as observed from our dataset. Some of the desirable traits of the compressed path are as mentioned below.

a. Informative nodes
b. Salient nodes
c. Proper order of appearance

The initial weight function IW (e_{ij}) is defined as follows.

$$IW(e_{ij}) = \frac{freq(i) + freq(j)}{\log(incd(i) + incd(j))} \text{..............} (1)$$

Where *freq* (*i*) and *freq* (*j*) are the number of edges connected to nodes *i* and *j* respectively and IW (e_{ij}) calculates the initial edge weights among adjacent nodes *i* and *j*; *incd (i)* and *incd (j)* are the number of incoming edges incident on the nodes *i* and *j* respectively.
The weight function defined in Equation (1) maintains the inclusion of strong grammatical compression links, i.e., it favors links between words that occur significantly in an order. However this does not guarantee that salient nodes will be included in the compressed path. To ensure that the compressed path also include salient words, we do a minor modification to our initial weight function to obtain our final weight function.

$$FW(e_{ij}) = \frac{IW(e_{ij})}{freq(i) \times freq(j)} \text{.................} (2)$$

The above weight function in Equation (2) ensures that our compressed path passes through the nodes with highest traffic or frequency as it reduces edge weights among important nodes, i.e. edges with greater frequencies. There is also another implied advantage of using this procedure which is Prepositional Phrase (PP) reductions. Pruning of PP attachments is an important aspect of summarizing sentences as the attachments often do not contain any relevant information, can be ambiguous at times and can be redundant as well. With our weight function we try to ensure that such spurious additions get clipped due to their lack of coherent order and low frequency. It is to be noted that the PP reduction method works only if such an attachment exists at the start or at the end of sentence and provided there is not enough high frequency nodes present in the vicinity. The word graph does not actively prunes the PP attachments; rather it is a structural and weighing function advantage. The pruning involving the prepositional phrases is by no means optimal; we have however discussed in the later section how we sought to rectify this using semantic constraints on the formation of sentences. Once we have calculated all the edge weights involved, we use K-shortest path algorithm to find the fifty shortest paths from the start node to the end node in the graph. Sentences of minimum 8 words length and containing at least a verb node are extracted for the summary. We re-rank all those paths by normalizing the total path weight over their lengths. The path with the minimum average edge weight is our compressed path. Our weighting function is such that frequency of nodes as well as association between the nodes directly correlates with a lower edge weight. This is easily understandable from the weight function as the denominator increases if the node has higher frequency. Thus the minimum edge weight paths are favored.

5.3 Syntactic and Semantic Modification

To create an informative summary it is always advisable to incorporate some language specific rules while generating the output sentences. If we solely depended on the word graph approach to create our summary it will suffer from grammatical incongruities at worst and redundant word representation at best. Grammatical incongruities may arise from the inclusion of a word pair which might not make any sense within the context but was nevertheless selected due to their high frequency. Redundant word representation although does not make the output summary incoherent it still degrades the overall quality of the summary.

Once we have generated the set of compressed sentences using the word graph, we use them as the input for the next module. Here, we create the constituency based parsed tree[7] for the compressed sentences. Observing and analyzing the constituency parse trees, we put some grammatical constraints on the word graph generated compressed sentences to obtain our final set of sentences for the summary output.

5.3.1 Parse Tree Modification

After sentence compression has been performed, we can work on the assumption that we have all the relevant information present in a given sentence; therefore, re-ordering and re-arranging in a grammatically coherent manner would not lead to any loss of information. We make a number of changes in the original structure of the parse tree based on its syntax and semantics. The changes are as follows.

1. Chains of conjuncts are split and each of them is attached to its parent node.
2. Synonyms (with the exception of stop words including determiners and common verb words) are merged into a single word.

[7] As part of the Stanford CoreNLP suite: http://nlp.stanford.edu/software/corenlp.shtml

3. Any prepositional phrase (PP) attachment appearing at the start of a sentence is dropped iff there exists no such production in the form of ***S→NP VP*** rooted under that PP expansion.
4. If two PP attachments are present such that one is rooted at a NP subtree and the other is at a VP node, then the PP attachment under VP is dropped provided
   i. The PP attachment rooted under a VP node is preceded by the PP at NP node in the sentence.
   ii. There does not exist any S production rooted under the PP subtree or VP subtree
   iii. There does not exist any verb phrase expansion under the PP subtree
5. Basic sentence formation constraint must be maintained i.e. no sentence can have less than 8 words and they must have at least one verb node.

Now we will proceed to show how some of the grammar rules we have outlined work. Let us take an example,

Example 1: *in recent times only few students are opting for STEM courses*

Its corresponding parse tree is given in fig 5.2. According to rule (3) stated above in section 5.3.1 since a PP attachment exists at the start of the sentence and no production of the form *S→NP VP* lies beneath it, the attachment "*in recent times*" will get pruned.

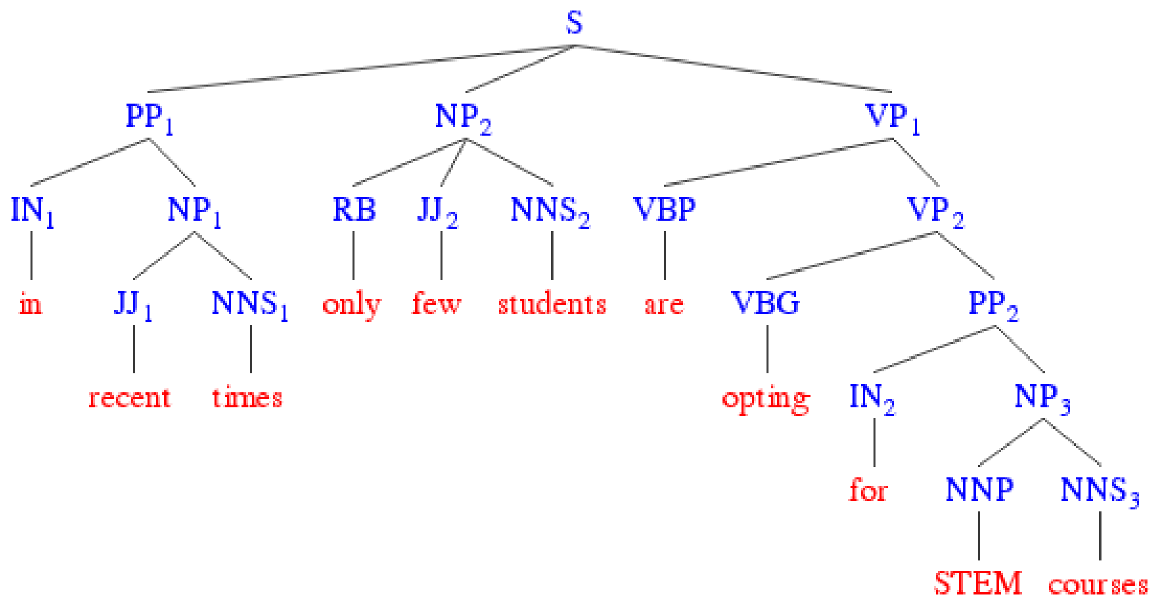

Figure 5.2: Parse Tree generated for the Example 1

Example 2: *the existence of gravitational wave was announced yesterday in the early morning*

The corresponding parse tree is given below in fig 5.3. According to rule (4) described above, the PP attachment "*in the early morning*" is pruned as another PP attachment under a NP node already precedes it

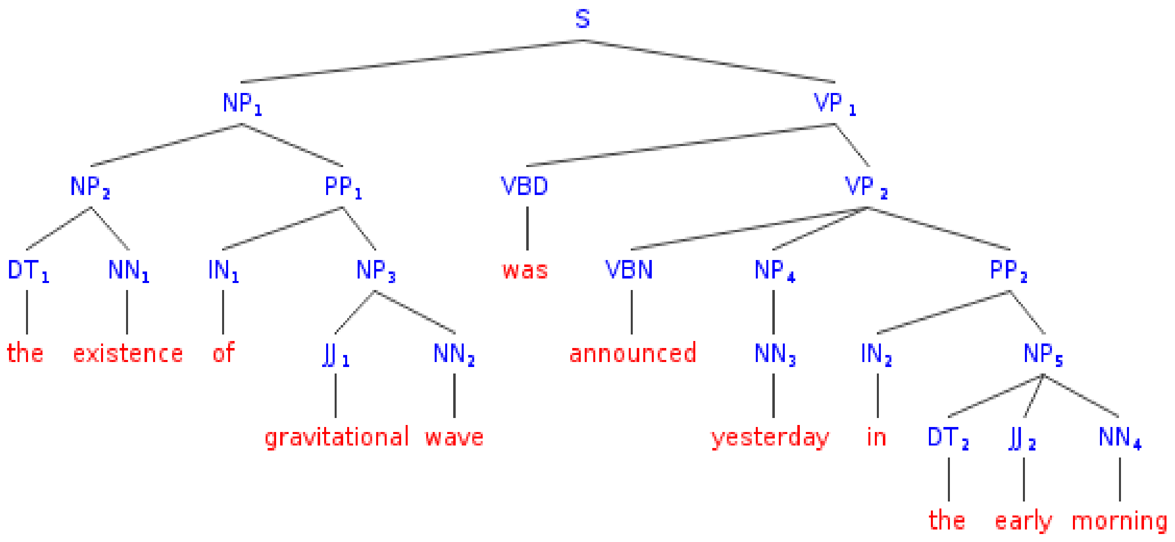

Figure 5.3: Parse Tree generated for the Example 2

Example 3: *the lecturer in his haste skimmed through the presentation to make his evening appointment*

The corresponding parse tree is given below in fig 5.4. Despite having a PP attachment under VP subtree, it cannot be pruned according to rule (4) since there exists a S production beneath the VP subtree.

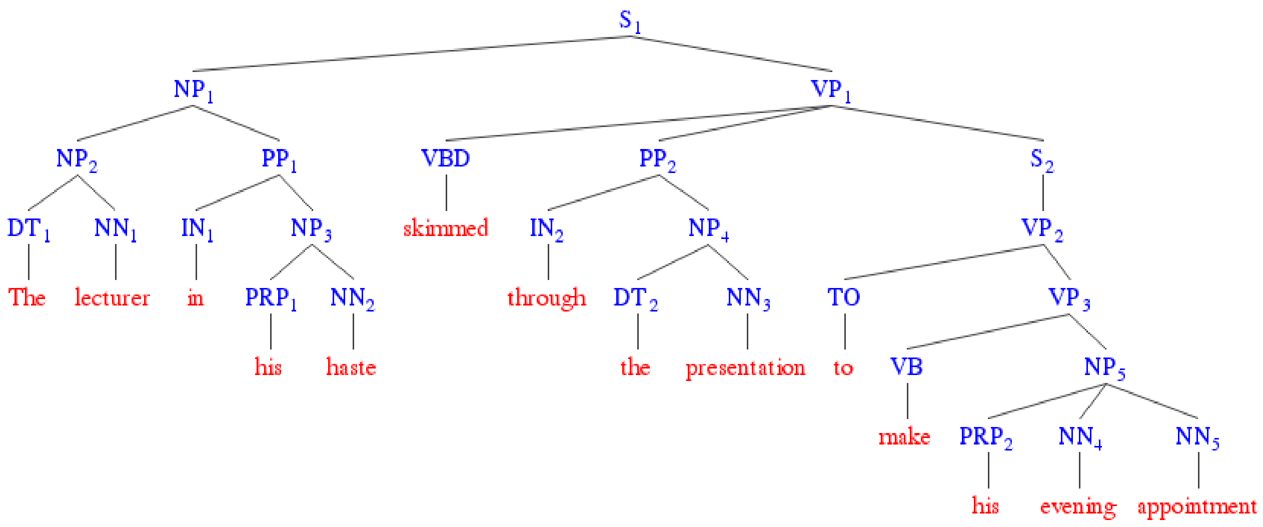

Figure 5.4: Parse Tree generated for the Example 3

5.3.2 Probabilistic Modification

Even the most well crafted grammar rules might become too generic to filter out redundancy. Therefore, as a final restriction we introduce a probabilistic constraint parameter to prune edges using conditional bigram probability. It might so happen that PP attachments, not covered under the previously discussed conditions, are still present in the generated sentences. In such cases, bigram probability of the preposition head term at the start of a PP attachment is calculated with respect to its previous term. Here the prepositional head term refers to the preposition node at which a PP subtree is rooted. If the conditional probability of a preposition head term occurring at a latter position down the sentence has a lower probability than that of a preposition term occurring at an earlier position within that sentence then the latter PP attachment is pruned, otherwise no pruning is carried out. The condition for this pruning is given as:

$$Pr(PP_l|prev\ term) < Pr\left(PP_f\middle|prev\ term\right)$$

Where *Pr* (*PP|prev term*) is the conditional probability of the preposition term at the start of a prepositional phrase given the previous word. PP_f and PP_l are the two preposition terms among which we make our comparison where PP_f refers to the preposition node which occurs earlier in the sentence and PP_l refers to the prepositional term occurring later in the sentence. This pruning follows the obvious logic that if such PP attachments are present in the sentence at such great depth it might be conveying important information. However, at the same time, given their position in the sentence we know that such attachments have persisted mainly because of their structure which allowed it to be retained so far into the process. Thus we reach a compromise by checking their probabilistic significance from a commonly used, versatile, annotated corpus to see how they fare.

We use the Manually Annotated Sub Corpus (MASC)[8] from which probability distribution was calculated. The MASC dataset was created from a portion of the Penn Treebank corpus, it has over half a million words in the form of written and spoken data including around 25,000 words divided among 19 sub-topics with the data annotated in 17 different part of speech tag types.

It is to be noted that all the above syntactic and semantic rules are not meant for universal application. Our summarization system mostly deals with formalized data as its input and as such all those data follow a particular grammatical structure. By studying and testing all the rules on our working data we have come to the conclusion of imposing them as they fit our work the best. For a different type of dataset, the application of the same rules might not yield in desired results.

5.4 Evaluation

It is difficult to evaluate abstractive summarization tasks objectively due to the lack of any concrete unified summary results. Both manual evaluation and automatic evaluation are constrained to their biases. Not every human generated summary will be the same and not all gold standard results are viable comparison samples. Nonetheless, we used the ROUGE 2.0 (Lin, 2004) metric for automatic evaluation, comparing our system generated summaries against human generated summaries. For human generated summaries we instructed three reviewers to construct summaries of the original input files for three compression rates of 90%, 80% and 85%.The purpose of this comparison is to test the system's performance at different compression rates. This evaluation results are presented in Tables 1−2. We used the

[8]http://www.anc.org/data/masc/corpus/

Opinosis[9] dataset as the gold standard for comparing our system against the MEAD[10]and Opinosis systems. Ganesan et al. (2010) proposed a graph based abstractive summarization system called *Opinosis[11]* in which the original input text is represented by a textual graph and the Opinosis system consequently explores this graph and scores the various sub-paths in the graph to generate the candidate summaries. The Opinosis dataset comprises extracted sentences from reviews on any particular topic. There are 51 such topics and the reviews on each topic provide the input text which needs to be summarized. The dataset also contains human generated summaries for the said topics which we use as the gold standard model summaries for comparison against our system generated summaries. MEAD (Radev et al., 2004) is a very well acknowledged extractive summarization tool which uses three features namely centroid, text position and sentence length to score and rank sentences to generate summaries. We use both the Opinosis and MEAD for a comparative evaluation. In particular the reason we chose Opinosis is because of its operational proximity to our proposed system. In addition to being one of the best abstractive systems till date, it also uses a mix of graph based approach and statistical approach which is quite similar to ours. While evaluating against the Opinosis gold standard dataset we limited our compression rate to 95% since most of the gold summaries ranged from 2 sentences to 4 sentences and thus our system generated summaries with 95% compression rate which corresponded length wise with the gold standard summaries in the Opinosis dataset. We took all the input documents present in the Opinosis dataset to serve as our test set and generated respective summaries. We evaluated the results generated by the Opinosis system and MEAD against the gold standard (cf. Table 3) for a better understanding of how our system performed in comparison to them.

Compression Rate	**Average Precision**	**Average Recall**	**Average F-Measure**
90%	0.9135	0.3831	0.5398
85%	0.8956	0.6573	0.7581
80%	0.8638	0.8185	0.8406

Table 1: Evaluation with Rouge-1 against Human Summaries

Compression Rate	**Average Precision**	**Average Recall**	**Average F-Measure**
90%	0.7576	0.3151	0.4451
85%	0.8161	0.5966	0.6893
80%	0.7956	0.7521	0.7732

Table 2: Evaluation with Rouge-2 against Human Summaries

[9]http://kavita-ganesan.com/dataset
[10]http://www.summarization.com/mead/
[11]http://kavita-ganesan.com/opinosis

ROUGE-1	**Average Precision**	**Average Recall**	**Average F-Measure**
Gold Standard	1	1	1
Our System	0.21	0.52	0.30
Opinosis	0.44	0.28	0.32
MEAD	0.10	0.49	0.15

Table 3: Evaluation with Rouge-1 against Opinosis Gold Standard

5.5 Results and Analysis

The evaluation results are presented in Tables 1−3.Results obtained by Rouge-1 and Rouge-2against the human generated summaries are presented in Table1 and Table 2 respectively, while Table3 presents the Rouge-1 scores obtained against the Opinosis gold standard. Evaluation results on machine generated summaries from the Opinosis and MEAD systems are also presented in Table 3 to provide a comparative understanding of how our system fares with respect to Opinosis and MEAD. For the results obtained against our human reviewers, 80% compression rate shows the most favorable results as opposed to a higher compression rate. At higher compression rates of 85% and 90%, there is a significant decrease in performance due to reduction in output size. This can be attributed to the fact that the human summaries were typically longer and match length wise with the system generated summaries obtained with 80% compression rate.
As far the evaluation against the gold standard is concerned, the proposed system outperformed the MEAD baseline by a significant margin, a 100% improvement in F-measure. Overall the Opinosis system performs best. Note that the Opinosis summaries ranged from 2 sentences to 4 sentences whereas our system generate summaries at desired compression rate and as we discussed earlier we had to set compression rate at 95%. And since our system is not optimal at such high compression rates, a slight dip in the F-measure in comparison to Opinosis' best performance is within acceptable limits. Therefore, overall our system can be considered comparable to the Opinosis system. Our system fails to produce better results at higher compression rates primarily because of a restriction imposed during the initial sentence compression stage i.e. maintaining a minimum of 8 words in every generated sentence. It fails to produce informative summaries at higher compression where shorter sentences with more selective set of words might have been able to convey the information more succinctly. We focus more on the content and structure of the generated sentences over individual isolated words. Therefore, at higher compression rates, our system produces fewer sentences while maintaining the word count as opposed to generating greater number of shorter sentences. This is a limitation of the proposed system which can be addressed by incorporating keyphrase extraction methods, supervised content selection technique, or tuning the optimal sentence compression rate, sentence length, etc. Furthermore if we look at the ROUGE scores presented in Table 3it can be observed that precision of our system is quite low compared to that of the Opinosis system. This is because our system's output were longer sentences with many spurious terms as opposed to theirs which were shorter and more relevant. Thus the fraction of relevant terms among the retrieved instances was lower for our system. However the data presented in table 1 and 2 shows a completely opposite picture where precision increases with higher compression. This is because the

human reviewers were instructed to maintain a minimum of 8 words in the summary sentences. In that scenario, precision was greater at higher compression rates since the fraction of relevant information among the retrieved instances were high. Despite the high precision, our system's F-measure was still low at higher compressions against the human summaries. Recall scores were low because a machine generated summary will always be less intuitive than a human generated one. Even at higher compression rates, the human summaries can contain the majority of relevant information by intuitively structuring the sentences and fusing multiple sentences while maintaining the 8 word restriction. This is however not true for the system which can alter sentence structure or prune parts of sentence in a much less intuitive manner. Hence the fraction of relevant instances that are retrieved will be lesser at higher compression rates since the output size gets smaller which causes lesser amount of relevant information to be retrieved. Thus in both the cases of our evaluation, system output degrades at higher compression.

6. Conclusions

We presented in this thesis a compression based text summarization method which uses a graph based technique to generate the sentence compression. It does not rely on any supervised technique or similarity measure for content selection; instead, it uses an efficient weighting function to obtain the important and salient nodes from a cluster of related sentences obtained from multiple documents. We then apply some syntactical rules to ensure that any more redundancies, if present, are eliminated. As a final step, we introduced a Probabilistic constraint imposing a stricter selection in grammatical structure. Sentence compression using word graph is performed only once for the entire dataset. However the syntactic and statistical constraints can get re-iterated until a desired compression rate is achieved by the output summary. Lastly, we presented in details the evaluation of our system and explained what its strengths and shortcomings are. Our approach takes a compromise between abstraction and extraction as we incorporate features from both the types. However, it must be noted that there is not any hard line where one stops and the other begins. The extraction and abstraction are intertwined and occur at every step of the process. We also did a chronological study of sentence fusion and summarization techniques and showed how our work was derived from past research and what the future of abstractive summarization holds for us.

7. Future Work

Abstractive text summarization has a lot of potential and new methods and techniques are being applied with promising results. As for our approach, this too has room for improvement. Filippova (2010) used her own version of scoring mechanism, while we used our own scoring method; it is always possible to come up with a better scoring function which is more inclusive of all the information and salience present within the sentence context. As for the syntactic and semantic components of an abstractive summarization system, it is not very practical to always depend upon hand-crafted rules as exceptions are always there and can quite significantly degrade system performance. Reliance on a more statistical approach to grammar rules also makes it liable to be dependent upon an external source for support, be it a training corpus or machine learning approach which might not always work in conjuncture with the input documents. The innate abstraction present in summaries makes the task of automatic abstractive text summarization considerably challenging to find a foolproof solution. In future we wish to improve our scoring algorithm which will lead to more informative summaries and also to incorporate more fine-grained phrase merging to enhance the grammatical texture of the output summaries, which in turn will help the system to perform better at higher compression rates.

8. Reference

Dragomir R. Radev, Eduard Hovy and Kathleen McKeown, 2002. Introduction to the Special Issue of summarization, Vol 28(4), pp 399-408, MIT Press.

Michael White, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce and Kiri Wagstaff, 2001. Multidocument summarization via information extraction. In Proceedings of the first international conference on Human language technology research, pp 1–7. Association for Computational Linguistics.

Pierre E.Genest and Guy Lapalme, 2012. Fully abstractive approach to guided summarization. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, Vol 2, pp 354–358. Association for Computational Linguistics.

Dan Gillick and Beniot Favre, 2009. A scalable global model for summarization. In Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing, pp 10-18. Association for Computational Linguisitcs.

Kevin Knight and Daniel Marcu, 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. Artificial Intelligence, Vol 139, pp 91-107. Elsevier.

Kathleen R. McKeown and Michel Galley, 2007. Lexicalized Markov grammars for sentence compression. In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pp 180–187, Rochester, New York, April. Association for Computational Linguistics.

Katja Filippova and Michael Strube, 2008. Dependency tree based sentence compression. In Proceedings of the 5th International Conference on Natural Language Generation, Salt Fork, Ohio, 12–14 June 2008, pp 25– 32.

Regina Barzilay and Kathleen R. McKeown, 2005. Sentence fusion for multi-document news summarization. Computational Linguistics, Vol 31(3), pp 297–328. Association for Computational Linguistics.

Katja Filippova, 2010. Multi-sentence compression: Finding shortest paths in word graphs. In Proceedings of the 23rd International Conference on Computational Linguistics, pp 322–330. Association for Computational Linguistics.

Florian Boudin and Morin Emmanuel, 2013. Keyphrase extraction for n-best re-ranking in multi-sentence compression. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pps 298–305, Atlanta, Georgia, June. Association for Computational Linguistics.

Hal Daume III and Daniel Marcu, 2004. Generic sentence fusion is an ill-defined summarization task. In Proceedings of the ACL-04 Workshop, pp 96–103, Barcelona, Spain, July. Association for Computational Linguistics.

Kapil Thadani and Kathleen R. McKeown, 2013. Supervised sentence fusion with single-stage inference. In Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 1410–1418, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Adam Berger and Vibhu O. Mittal, 2000. Query-relevant summarization using FAQs. In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp 294-301. Association for Computational Linguistics.

Chiori Hori and Sadaoki Furui, 2003. A new approach to automatic speech summarization. Multimedia, IEEE Transactions, Vol 5(3), pp 368-378. IEEE.

Hal Daume III and Daniel Marcu, 2006. Bayesian query-focused summarization. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp 305-312. Association for Computational Linguistics.

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

Dragomir R. Radev, 2000. A Common Theory of Information Fusion from Multiple Text Sources Step One: Cross-Document Structure. Proceedings of the 1st SIG dial workshop on Discourse and dialogue, Vol 10. Association for Computational Linguistics.

William C. Mann and Sandra A. Thompson, 1988. Rhetorical Structure Theory: Toward a functional theory of text organization. Vol 8(3). pp 243-281.

Randall H. Trigg and Mark Weiser, 1986. TEXTNET: a network-based approach to text handling. ACM Transactions on Information Systems (TOIS), Vol 4 (1), pp 1-23. Association of Computing Machinery.

Gerard Salton, Amit Singhal, Mandar Mitra and ChrisBuckley, 1997. Automatic Text Structuring andSummarization, Information Processing andManagement, Vol 33 (2), pp 193—207.

Dragomir R. Radev and Kathleen R. McKeown, 1998.Generating natural language summaries from multiple on-line sources. Computational Linguistics, Vol 24 (3), pp 470-500. MIT Press.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert, 1996. Description of the UMass System as Used for MUC-6. In Proceedings of the Sixth Message Understanding Conference (MUC-6), pp 221—236.

James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang, 1998. Topic detection and tracking pilot study: final report. In Proceedings of the Broadcast News Understanding and Transcription Workshop.

Marti Hearst, 1994. Multi-Paragraph Segmentation of Expository Text. In Proceedings of the 32ndAnnual Meeting of the Association for Computational Linguistics, pp 9-16. Association for Computational Linguistics.

Min-Yen Kan, Judith L. Klavans and KathleenMcKeown, 1998. Linear segmentation and segmentrelevance. In *Proceedings of 6th InternationalWorkshop of Very Large Corpora (WVLC-6)*, pp 197-205.

James Allan, 1996. Automatic hypertext link typing. In Proceedings ofthe Seventh ACM Conference on Hypertext, pp 42-52. Association for Computing Machinery.

Regina Barzilay and Michael Elhadad, 1997. Using lexical chains for text summarization. In Proceedings of the ACL workshop on intelligent scalable text summarization, Vol 17, pp 10–17. Association for Computational Linguistics.

Regina Barzilay and Noemie Elhadad, 2003. Sentence alignment for mono-lingual corpora. In Proceedings of the 2003 conference on Empirical methods in natural language processing, pp 25-32. Association for Computational Linguistics.

Kathleen R. McKeown, 1985. Text generation: using discourse strategies and focus constraints to generate natural language text. Cambridge University Press.

Robert E Schapire and Yoram Singer, 2000. BoosTexter: A boosting-based system for text categorization. Machine Learning, Vol 39(2), pp 135-168. Springer Netherlands.

Stuart Shieber and Rani Nelken, 2006.Towards robust context-sensitive sentence alignment for monolingual corpora. In Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics, pp 161–168. Association for Computational Linguistics.

Regina Barzilay and Lillian Lee, 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In Proceedings of HLT/NAACL, pp 16-23. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau, 2004. TextRank: Bringing order into texts. In Proceedings of EMNLP 2004, pp. 404-411. Association for Computational Linguistics.

Regina Barzilay, Kathleen R. McKeown and Michael Elhadad, 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, pp 550-557. Association for Computational Linguistics.

Regina Barzilay, Michael Elhadad and Kathleen R. McKeown, 2002. Inferring strategies for sentence ordering in multi-document news summarization. Journal of Artificial Intelligence Research, Vol 17, pp 35–55. AI Access Foundation.

Vasileios Hatzivassiloglou, Judith L Klavans, Eleazar Eskin, 1999. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In Proceedings of the 1999 joint sigdat conference on empirical methods in natural language processing and very large corpora, pp 203-212.

Vasileios Hatzivassiloglou, Judith L Klavans, Melissa L Holcombe, Regina Barzilay, Min-Yen Kan and Kathleen McKeown, 2001. Simfinder: A flexible clustering tool for summarization. In Proceedings of the Workshop on Summarization in NAACL-01.

Kathleen R McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L Klavans, Ani Nenkova, Carl Sable, Barry Schiffman and Sergey Sigelman, 2002. Tracking and summarizing news on a daily basis with Columbia's Newsblaster. In *Proceedings of the Human Language Technology Conference (HLT-02)*, pp 280–285.

Igor Melcuk, 1988. Dependency Syntax: Theory and Practice. The SUNY Press, pp 428.

George A. Miller, Richard Beckwith,Christiane Fellbaum, Derek Gross and Katherine J. Miller, 1990. Introduction toWordNet: An online lexical database.*International Journal of Lexicography*, Vol 3(4), pp 235–245.

Michael Collins, 1996. A new statistical parser based on bigram lexical dependencies. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Assocation for Computational Linguistics.

Inderjeet Mani and Eric Bloedorn, 1997. Multi-document summarization by graph search and matching. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-97), pp 622-628, Providence, Rhode Island. AAAI.

Kavita Ganesan , ChengXiang Zhai and Jiawei Han, 2010. Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. In Proceedings of the 23rd International Conference on Computational Linguistics, pp 340–348, Beijing.

Chin-Yew Lin and Eduard Hovy, 1998. Automatic Text Summarization in SUMMARIST, *TIPSTER '98* .In Proceedings of a workshop on held at Baltimore, Maryland,pp 197-214. Assoication of Computational Linguistics.

Christaine Fellbaum, Collin Baker, Nancy Ide and Rebecca J. Passonneau, 2012. The MASC Word Sense Sentence Corpus. In Proceedings of the Eighth Language Resources and Evaluation Conference. LREC.

Christopher D. Manning, David McClosky, Jenny Finkel, Mihai Surdeanu, John Bauer and Steven J. Bethard, 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp 55-60. Assoication of Computational Linguistics.

Kevin Knight and Daniel Marcu, 2000. Statistics based Summarization-step one: Sentence compression. American Association for Artificial Intelligence, Vol 139, pp 91-107. Elsevier.

Steven Bird,Ewan Klein and Edward Loper, 2009. Natural Language Processing with Python. O'Reilly Media Inc.

Jackie C. K. Cheung and Gerald Penn, 2014. Unsupervised sentence enhancement for automatic summarization. *EMNLP* pp 775-786. Assoication of Computational Linguistics.

Hongyan Jing and Kathleen R. McKeown, 2000. Cut and Paste based text summarization. In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference.Assoication of Computational Linguistics.

Jade Goldstein, Jaime Carbonell,Vibhu Mittal and Mark Kantrowitz,1997. Multi-document summarization by sentence extraction.InProceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization, Vol 4, pp 40-48. Assoication of Computational Linguistics.

James Clark and Mirella Lapata, 2008. Global inference for sentence compression: An integer linear programming approach. Journal of Artificial Intelligence Research, pp 399-429.

Joel L.Neto, Alex A. Freitas, Celso A. A. Kaestner, 2002. Automatic text summarization using a machine learning approach.Advances in Artificial Intelligence, pp 205-215. SBIA.

Kathleen McKeown and Min Y Kan, 2002. Corpus trained text generation for summarization. In Proceedings of the Second International Natural Language Generation Conference, pp 1-8.

Lidong Bing, Piji Li, Rebecca J. Passonneau, Wai Lam, Weiwei Guo, Yi Liao, 2015. Abstractive Multi-Document Summarization via Phrase Selection and Merging , pp 1587-1597. Assoication of Computational Linguistics.

Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ILP for extractive summarization. InProceedings of ACL.

Chin-Yew Lin, 2004. Rouge: A package for automatic evaluation of summaries. In Proceedings of the workshop of Text Summarization branches out, Vol 8. ACL-04.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu and Dan Jurafsky. 2013. Deterministic coreferenceresolution based on entity-centric, precision-ranked rules. Computational Linguistics, Vol 39(4), pp 885–916.

Laura Kassner, Vivi Nastase & Michael Strube, 2008. Acquiring a taxonomy from the German Wikipedia. In Proceedings of the 6th International Conference on Language Resources and Evaluation, Marrakech, Morocco.

Michael Strube and Simone Paolo Ponzetto, 2006. WikiRelate! Computing semantic relatedness using Wikipedia. In Proceedings of the 21st National Conference on Artificial Intelligence, pp 1419–1424.