Abstract. The need for text summarization is crucial as we enter the era of information overload. However, the current implementations are specific to a domain or a genre of the source document. In this paper, we discuss an algorithm for text summarization which is independent of domain and document source. This algorithm creates text summaries by analyzing the logical structure of the sentences. Sentences are parsed and important relationships are identified, stored in the form of a graph, thus graph corresponding to each sentence in the document is generated and merged to form graph of the document, now this graph is clustered into sub-graphs which represent the different topics in the document. Then a graph scoring algorithm scores the graph, and helps in extracting the important sentences towards summary. To increase the coherence of the summary, the pool of extracted sentences undergoes some transformation in a specified order, resulting in final sentences that form the summary of the document.

1 Introduction

A huge amount of on-line information is available on the web, and it is still growing. Even search engines output a large number of documents for a user's query. It is difficult for the user to find out the one he actually needs, because most of the naive users are reluctant to make the cumbersome effort of going through each of the documents. So systems that can automatically summarize one or more documents are becoming increasingly desirable.

A summary can be loosely defined as a text that is produced from one or more texts. Automatic summarization is to use automatic mechanism to produce a finer version for the original document. Spark-Jones [17] discussed several ways to classify summaries. The following three factors are considered to be important for text summarization.

- **Input factors**: text length, genre, number of documents
- **Purpose factors**: audience, purpose of summarization.
- **Output factors**: running text or headed text etc.
Summaries can be classified into different types based on dimensions, genre, and context.

- **Dimensions**: Single vs. Multi-document summarization
- **Genre**: Headlines, outlines, minutes, chronologies, etc.
- **Context**: Generic, Query specific summaries

As pointed out in Mani and Maybury [12], summaries can be classified into extracts (most relevant sentences are selected from the text), and abstracts (text is analyzed, a conceptual representation is provided which in turn is used to generate sentences that form summary). Conventional text summarization systems produce summaries by using sentences or paragraphs as basic unit, giving them degree of importance [3], sorting them based on the importance, and gathering the important sentences.

In this paper we have proposed an extract type summary generation, and the initial experiments show encouraging results and we are in the process of completing a robust implementation of the system. We believe this approach can be extended to produce abstracts as well.

### 2 Background

Most of the summarization work done till date is based on extraction of sentences from the original document. The sentence extraction techniques compute score for each sentence based on features such as position of sentence in the document [1,3], word or phrase frequency [10], cue phrases (terms which indicate the importance of the sentence towards summary e.g. “this article talks about”) [3], Lexical chains [11], key phrases [2,7], LexPageRank [4]. There were some attempts to use machine learning (to identify important features), use natural language processing (to identify key passages or to use relationship between words rather than bag of words). The application of machine learning to summarization was pioneered by Kupiec, Pedersen, and Chen [8], who developed a summarizer for scientific articles using a Bayesian classifier.

To produce an effective summary, one has to really understand the point of a text. This requires semantic analysis, discourse processing, and sentence generation. This system is an attempt in that direction. Here we first try to understand, and mark the topics in the text, with the help of its logical representation. Then the important pieces of information are selected and used in generation of summary.

### 3 System Description

The architecture of the system is shown in the Fig 1. The system has both text analysis component and the summary generation component. The text analysis part is a syntactic analysis component followed by a component which does a logical analysis for each sentence. Text normalization is syntactic analysis of the text which includes extracting the text from the document (format conversion, if needed), removing floating objects like figures, tables, identification of titles and subtitles, and dividing the text into sentences. The logical analysis includes anaphora resolution, extracting the
named entities, followed by a dependency analysis of a sentence which gives the logical connection between the words. The logical analysis produces a graph with the words in their root form as nodes and these nodes are joined by the way of their relationship between them. The summary generation component identifies important pieces of information from the graph using graph clustering, and graph scoring and thus helps in identification of the important sentences in the document.

![Architecture of the system](image)

**Fig 1:** Architecture of the system

### 4 Text Analysis

We try to identify the underlying conceptual representation of the document. This involves both syntactic and semantic analysis. We assume that the input document
can be of any form, so the system first applies document converters to extract the text from the input document. We have pdftotext, doctotext, ppttotext, pstotext and htmltotext converters, available with us.

4.1.1 Text Normalization

The text normalization is a rule based component which removes the unimportant objects like figures, tables, identifies the headings and subheadings and handling of non-standard words like web urls and emails and so on. The text is then divided into sentences for further processing.

4.1.2 Sentence Marker

This module divides the document into sentences. At first glance, it may appear that using end-of-sentence punctuation marks, such as periods, question marks, and exclamation points, is sufficient for marking the sentence boundaries. Exclamation point and question mark are somewhat less ambiguous. However, since these forms of punctuation are not used exclusively to mark sentence breaks, sentence boundaries are ambiguous. The sentence marker considers the following three features for any complete sentence. [15]

1. Every sentence has a verb
2. Every sentence starts with an uppercase letter
3. All proper nouns start with an uppercase letter

4.2 Logical Analysis & Parsing of Sentences

This module analyzes the sentence structure with the help of available NLP tools like anaphora resolution, named entity extraction and sentence parser. Some of these tools are optional while the others are necessary.

Anaphora resolution is a rule-based component which resolves pronoun referents based on some preference. As in Lappin and Leass[9], a score is computed for each candidate antecedent according to a number of factors, including whether it precedes or follows the anaphor, distance between it and the anaphor, agreement. After calculating the score, the system attempts to find the most likely antecedent for referential pronouns from the current sentence and preceding sentences.

The general named entity extractor can identify named entities (persons, locations and organizations), temporal expressions (dates and times) and certain types of numerical expressions. This named entity extractor uses both syntactic and contextual information. The context information is identified in the form of POS tags of the words and used in the named entity rules, some of these rules are general and while the rest are domain specific.

While anaphora resolution, and named entity extraction are optional, the parsing of sentence is necessary. Typically, all the information retrieval systems assume that most of information in a document is provided by the nouns, but verbs and their relationship between other words in a sentence are also needed to produce an effective summary. This idea is supported by LAKE keyword extraction based summarizing
system[2], their system performed well when they included verb form in their candidate patterns. There could be many relations possible between the words in a sentence, but we feel that the relations between verb and targeted words (such as subject, object, noun-noun modifier, verb-verb modifier, and named entities with in that sentence), are sufficient to extract the meaning of a sentence, and hence help the summarization. We refer to these relationships as target relationships throughout the paper. The system uses Collins parser [13,14] to identify the target relationships. Collins parser creates the parse tree for a sentence based on a generative, lexicalized and probabilistic parsing model. Fig 2.a shows an example sentence and the corresponding output generated by Collins parser.

**Example sentence:** Verbs also differ in the number of arguments that they take based on the context.

```
(TOP~differ~1~1 (S~differ~3~3 (NP~Verbs~1~1 (NPB~Verbs~1~1 Verbs/NNP ) ) (ADVP~also~1~1 also/RB ) (VP~differ~2~1 differ/VBP (PP~in~2~1 in/IN (NP~number~2~1 (NPB~number~2~2 the/DT number/NN ) (PP~of~2~1 of/IN (NP~arguments~2~1 (NPB~arguments~1~1 arguments/NNS) (SBAR~take~1~1 (S~take~2~2 (NP~they~1~1 (NPB~they~1~1 they/PRP)) (VP~take~2~1  take/VBP (VP~based~2~1 based/VBN (PP~on~2~1 on/IN (NP~context~1~1 (NPB~context~2~2 the/DT context/NN))))) ))))))
```

**Fig 2.a:** Output of Collins parser:

**Fig 2.b:** Output of the Graph Generator

**Fig 2:** a) The first part shows the output produced by the Collins parser b) second part is the graph generated from the output of the Collins parser

### 4.3 Graph Generation

Now the graph generator uses this output and some linguistic rules to identify the target relationships (such as subj, obj ...). This extracted information is represented in the form of graph, with the words in their root form as the nodes and these nodes are joined by the way of their relationship between them. A relation between two nodes can be observed more than once, so we will keep track of the number of times the relation has occurred. Figure 2.b shows an example of such a graph. Now graphs cor-
responding to two different sentences can be merged using the nodes that are common to the two graphs and the new arcs being the union of the old arcs. The graphs corresponding to all the sentences in the input document are generated and merged to form a single graph, called document graph.

A document may talk about a main topic and some sub topics, these sub topics add some information to the main topic, i.e. the reference of main topic can be found in all these individual sub topics. A sub topic is highly related within itself, and sparsely related to other sub topics. So the document graph will not be uniformly distributed, i.e. some nodes are highly related within themselves and less linked to other nodes. So the graph needs to be clustered in order to identify the different sub topics in the document. In this way we can efficiently identify the topics even though they are randomly distributed in the input document.

5 Summary Generation

The task of summary generation component is to identify main topic and sub topics, select important pieces of information from the graph, and extract the corresponding sentences from the document, order them and transform these sentences until the desired summary length has been reached. A graph clustering algorithm will cluster the graph into sub graphs which correspond to the sub topics in that document, followed by a graph scoring algorithm which will help in determining the important pieces of information.

5.1 Topic Clustering

Graph clustering is the separation of sparsely connected dense graphs from one another. A graph clustering algorithm will divide the document graph into sub graphs. These sub graphs will denote the different sub topics in that document. Graph clustering algorithms like minimum-cost spanning trees clustering, MacQueens k-means clustering, Maximum-Cut Clustering can be used [5] to identify the topic boundaries in the graph.

A link between two sub graphs indicates the relation between the two topics which represent the sub graphs. It is generally agreed upon that the main topics of a text are signaled by terms that occur throughout the text, while sub topics are signaled by terms that aggregated in limited passages [6]. Hence the sub graph which is linked to more number of other sub graphs is referred as the main topic of the document. The other sub graphs are considered as sub topics. Finally after topic clustering we will get the different sub graphs which denote different topics the text is talking about.

5.2 Graph Scoring using PageRank

Since the graph constructed is analogous to the web graph, we use the popular pageranking algorithm[16] to score the graph. Pageranking algorithm treats a directed edge between the nodes A and B as a recommendation made by A to B, so the algorithm adds some part of A's weight to B. Finally a node which is linked to more number of
nodes will get high score, and is treated as important. The important pieces in a graph are triples with a highly weighted node together with the most highly weighted of its neighbors and the relationship between them. We will select top 'n' of such triples (the value of n depends on the desired summary length). These triples are separated as main and sub triples, depending on, whether a triple belongs to main topic or sub topic.

5.3 Final Sentence Selection

The graph scoring module identifies important triples from the document graph. Now the document is scanned to identify the sentences which talk about the selected main triples, and sentences which connect the main topic with the other sub topics. The later is supported by the observation that the sentences which connect the main topic and sub topics contribute more towards summarization than the sentences which actually talk about the sub topic.

To increase coherence of the summary the following transformations were used, in the specified order:

- Add sentences to the pool so as to avoid dangling discourse relations. For example if a sentence starts with “afterwards” or “but”, the preceding sentence was marked as important as well and added to the set of important sentences.
- Some sentences are removed depending on the length of the desired summary. If a short length summary is requested, than it is good to select many short sentences and remove very long sentences. If the length of summary is comparable with the length of the document than sentences which are less than some threshold are removed from the pool.
- Remove questions, title and subtitles from the set of sentences.
- Rewrite sentences by deleting marked parenthetical units.
- Each third person pronoun that referred to an entity that was not mentioned already in the summary was replaced with the complete referring expression, if previously computed.

In the final step, we generate the summary by concatenating the remaining sentences.

6 Conclusion

In this paper, we proposed an algorithm which will represent the document as a graph and then use a graph clustering algorithm to divide the document into sub topics. The accuracy of the sub topics identified will depend on the components of the graph, i.e. the way in which the graph is generated. While generating the summary, we give more preference to the sentences which join the main topic and sub topics, than the sentences which talk about the sub topics. Because they will improve the efficiency and coherence of the summary. The initial experiments show encouraging results and we are in the process of of completing a robust implementation of the system.

We are using many NLP tools that are available in generating the graph. While anaphora resolution, and named entity extraction are not really mandatory, they are included to increase the efficiency of the system.
References