Extractive Based Automatic Text Summarization

Sagar M. Patel[1], Vipul K. Dabhi[2], Harshadkumar B. Prajapati[2]

[1] Department of Information Technology, Chandubhai S. Patel Institute of Technology (CSPIT), Charotar University of Science and Technology (CHARUSAT), Changa, Gujarat, India.
[2] Department of Information Technology, Faculty of Technology, Dharmsinh Desai University (DDU), Nadiad, Gujarat, India.

Corresponding author. Tel.: +918866493361; email: sagarmpatel.it@gmail.com
Manuscript submitted May 7, 2016; accepted July 17, 2016.
doi: 10.17706/jcp.12.6.550-563

Abstract: Automatic text summarization is the process of reducing the text content and retaining the important points of the document. Generally, there are two approaches for automatic text summarization: Extractive and Abstractive. The process of extractive based text summarization can be divided into two phases: pre-processing and processing. In this paper, we discuss some of the extractive based text summarization approaches used by researchers. We also provide the features for extractive based text summarization process. We also present the available linguistic preprocessing tools with their features, which are used for automatic text summarization. The tools and parameters useful for evaluating the generated summary are also discussed in this paper. Moreover, we explain our proposed lexical chain analysis approach, with sample generated lexical chains, for extractive based automatic text summarization. We also provide the evaluation results of our system generated summary. The proposed lexical chain analysis approach can be used to solve different text mining problems like topic classification, sentiment analysis, and summarization.

Key words: Automatic text summarization, extractive, abstractive, linguistic processing, lexical chain analysis, topic classification, sentiment analysis.

1. Introduction

In today's era of computerized world, the information grows at an exponential rate and it is very difficult to find the representative summary of available information. Automatic text summarization technique finds the relevant information from document or text. It reduces the time required for reading the whole document and also reduces the space needed for storing large amount of data. The automatic text summarization problem can be divided into two sub-problems, Single Document and Multi-Document [1]. The input of the single document summarization process is a single document and output is summary for the given document [2]. The multi-document summarization process takes multiple documents related to a single topic and generates summary by considering important sentences from all documents [3].

A summary can be generated by selecting some part of original text document or can be generated by including all relevant information of the text document [2]. However, both cases provide a summary which reduces the reading time. Automatic text summarization problem can be solved using two approaches: Extractive and Abstractive [1]. An extractive text summarization extracts important sentences from the original document. Extractive approach uses statistical or linguistic features for selecting informative sentences. An abstractive text summarization recognizes the original text and re-tells it in fewer words.

Abstractive summarizer first understands the main concepts of document and expresses those concepts in fewer words. It uses linguistic features to infer the text and provide the same meaning by generating shorter text. The generation of the shorter text is the part of the Natural Language Generation (NLG).

In 1969, new methods such as cue-phrase, title in document, and location of sentence was introduced in automatic extraction [4]. The trainable document summarizer was developed in1995 by J. Kupiec *et al*. [5] which provides more extractive features like sentence length and uppercase letters. The linguistic preprocessing: parsing was modified in 2003 by D. Kelin *et al*. [6]. They provided Probabilistic Context-Free Grammars (PCFG) which is more accurate than previous version. Three document operations: comparison, categorization and scrutinisation were proposed in XML based architecture for document validation by Prajapati and Dabhi in 2010 [7].

There are no pure abstractive approaches to solve text summarization problem, but some researchers tried to solve it in abstract way [2]. There are several abstract based approaches like statistics based, cut and paste approaches, and sentence reduction approach. Abstract approach uses syntactic knowledge, statistics (computed using corpus), and context information (computed by information extraction). The abstract based summarization is quite difficult to generate because of machines limitation and also it's a part of Natural Language Generation. An extractive summarization process can be decomposed into two steps: preprocessing and processing. Preprocessing is the linguistic information extraction from the given text or documents. Processing is the next step in which it will find the important information using statistics or linguistic features [8]-[10].

The rest of the paper is structured as follows. Section 2 presents linguistic preprocessing with example. It also discusses tools useful for automatic text summarization. Section 3 describes different extractive based approaches used for automatic text summarization. Section 3 also provides some of the measures used for comparing extractive text summarization. Section 4 provides the information about the parameters and tools for evaluation of generated summary. Section 4 provides survey on lexical chain analysis and also provides our proposed lexical chain approach for extractive based automatic text summarization. It also provides the generated sample results of lexical chains and also compares result of proposed system generated summary with human generated summary. Conclusions are presented in Section 5.

2. Linguistic Preprocessing for Automatic Summarization

In this section, we describe the linguistic preprocessing for text processing. Section 2.1 provides the different linguistic processing steps and Section 2.2 provides the available tools/plugins for linguistic preprocessing.

2.1. Linguistic Preprocessing

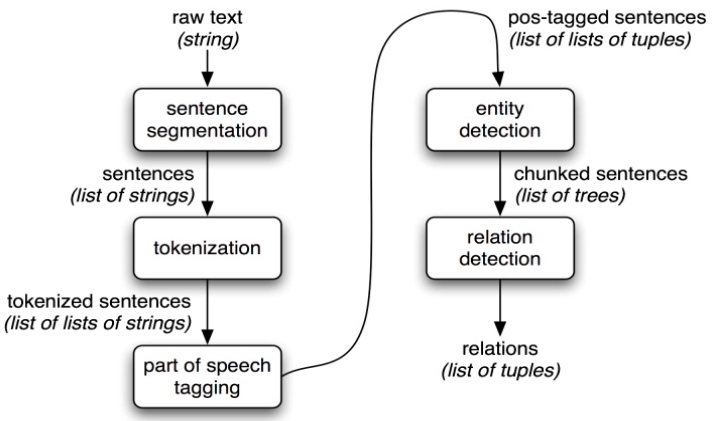

Fig. 1. Simple pipeline architecture for an information extraction system [11].

Linguistic preprocessing is an initial step for generating structural representation of the document. Fig. 1 shows the pipeline architecture of an information extraction process which is available in Natural Language Processing with python [11].

2.1.1. Sentence segmentation

To manipulate text at the word level, the prerequisite is to divide the text into individual sentences. Sentence segmentation [11] converts raw text into sentences, as a list of strings. Brown corpus provides the accessibility at sentence level. Following example shows the input text in form of a string and output produced by segmentation step in terms of a list of strings (segments). The segments are extracted as a list of sentences from the given document or text.

Input Text: John owns a car. It is a Toyota.

Output: Segm1: John owns a car. Segm2: It is a Toyota.

2.1.2. Tokenization

Tokenization [12] identifies the word tokens from given sentence. Tokenization takes a list of sentences as an input and provides a list of tokens as output. Following example shows the input and output of tokenization process. Here, the tokens are the words extracted from given sentence, separated by comma.

Input: John owns a car.

Output: [[John], [owns], [a], [car], [.]]

2.1.3. Part of speech tagging (POS Tagging)

A part of speech tagger or POS tagger [11] analyses output of Tokenization or sequence of words, and assigns appropriate part of speech tag to each word. POS is useful in extraction of nouns, adverbs, adjective, which provide some meaningful information about text. POS tagging takes a list of tokens as input and generates a list of tuples with POS annotation. Following example shows that the POS tagger takes an input in terms of tokens and provides the output with POS annotation. In the example, NP stands for noun phrase, NNP means noun (proper, singular), VP stands for verb phrase, VBZ indicates verb in present tense (3rd person singular), DT stands for determiner, and NN indicates noun (common, singular or mass).

Input: [[John], [owns], [a], [car], [.]]

Output: (NP (NNP John)) (VP (VBZ owns) (NP (DT a) (NN car))) (. .)

2.1.4. Entity detection

Entity detection provides the identification of predefined categories such as person, location, quantities, percentage, organizations, expressions of times, monetary values, etc [13]. The system called Name Entity Recognition (NER) provides the entity detection linguistic processing. NER system uses linguistic grammar-based techniques and also statistical model to identify the entity. Entity detection takes POS annotation tuples and provides the output in form of a tree. Here, the input is the POS annotation tuples and provides the output with NER values such as person, organization, etc.

Input: (NP (NNP John)) (VP (VBZ owns) (NP (DT a) (NN car))) (. .)

Output: John->Person

2.1.5. Relation detection

Relation detection identifies the possible relation between two or more chunked sentences. One of the features, co-reference words, provides the link between pronouns and its corresponding nouns [14]. Co-reference chain provides a relation between two or more sentences. Co-reference is useful in replacement of the pronouns with proper nouns. Relation detection step takes a list of tree of parsed string as input and provides the list of tuples as output. The following example takes the input of parse tree and gives the relation between both sentences.

Input Text: John owns a car. It is a Toyota. (In form of parse tree)

Output: "a car" -> "a Toyota"; "It" -> "a Toyota"

2.2. Tools/Plugins for Linguistic Preprocessing

Table 1 provides a list of linguistic processing components with their respective tools and plugins. The listed tools/plugins are explained in this sub-section with their features.

2.2.1. Stanford CoreNLP

Stanford CoreNLP [15] provides a set of natural language analysis libraries with API. Stanford CoreNLP takes a raw English text as input and it gives the base forms of words, their POS, NER, and indicates which noun phrases refer to the same entities. The Stanford CoreNLP code is an open source and Java based. Stanford CoreNLP is licensed under the GNU General Public License.

Table 1. Tools/ Plugins for Different Preprocessing Tasks

Sr. No.	Preprocessing Components	Tools / Plugins			
		Stanford CoreNLP	NLTK	OpenNLP	Rapid Miner
1.	Tokenization	✓ (Tokenizer)	✓	✓	✓
2.	POS tagging	✓ (POS Tagger)	✓	✓	✓ (Filtering)
3.	Stemming/ Lemmatization	✓ (Lemma)	✓	✓ (Stemmer)	✓
4.	N-Gram/Bi-Gram	✓ (Full)	--	✓	✓
5.	Parsing	✓ (Parser)	✓	✓	--
6.	Name Entity Recognition	✓ (Full)	✓	✓	--
7.	Co-Reference	✓ (Full)	--	✓	--

2.2.2. Apache OpenNLP

Apache OpenNLP [16] is a machine learning based toolkit for the natural language processing. Apache OpenNLP supports all preprocessing steps listed in Table 1. The source code of Apache OpenNLP is publicly available and is written in Java.

2.2.3. Natural language toolkit (NLTK)

NLTK [17] is python based toolkit for natural language processing. It provides easy to use interface for over 50 corpora/dictionary and lexical resources such as WordNet. NLTK is licensed under the Apache NLTK. NLTK provides almost all NLP tasks except Relation Detection and N-gram/Bi-gram models.

2.2.4. Rapid miner-text processing extension

Rapid Miner [18] provides text processing extension in which, it provides some basic preprocessing tasks such as tokenization, stemming, transformation, and filtering. Usage of Rapid Miner is preferred when a supervised learning technique is used to solve the text summarization problem.

3. Processing in Extractive Summarization

In this section we provide a survey of processing phase of extractive based text summarization. In this section, Section 3.1 presents a survey of different techniques used for extractive based approaches for automatic text summarization. Section 3.2 provides features used by these techniques for extractive based text summarization problem.

3.1. Technique used for Extractive Based Text Summarization

The main goal of extractive based text summarization is picking out most informative sentences from the given document. Next, we present different techniques used by researchers for extractive based text summarization.

3.1.1. Neural networks

There are three steps in neural network based text summarization process: 1) neural network training, 2) feature fusion, and 3) sentence selection [19]. The first step provides training to neural network for recognition of the type of sentences that should be considered as summary sentences [19]. The feature

fusion step has two sub steps: 1) pruning the neural network and 2) collapsing the effects of common features. The feature fusion step generalizes the important features, which must remain in the summary sentences. The sentence selection uses the improved neural network to filter the text and also for selection of only highly ranked sentences. The sentence selection step also controls the extraction of the important summary sentences [19].

3.1.2. Regression to estimate feature values

Mathematical regression [20] could be a good technique for estimation of the text feature weights. The mathematical function can relate the output with the input. Regression process takes many manually summarized documents with independent input variable and also with corresponding outputs as its training data. Then, it will generate the relation between input and output. The next step is to provide the testing data to the model for evaluation of its efficiency.

3.1.3. Conditional random fields (CRF)

Sometimes humans create the document summary by considering the summarization as a sequence labelling problem. CRF [21] provides the label sequence 1 for summary sentence and 0 for non-summary sentence. CRF provides the labels to the sentences such that the likelihood of the whole sentence sequence is maximized. CRF provides a probabilistic framework, which has two steps: parameter estimation and feature consideration. Parameter Estimation: set of weights in a CRF is usually estimated by a maximum likelihood procedure, by maximizing the conditional log-likelihood of the labelled sequences in the training data. Various methods can be used to optimize the likelihood, including Iterative Scaling algorithms. Feature Consideration: the basic features commonly used in CRF are sentence position, sentence length, log likelihood, uppercase words, and similarity to neighbouring sentence. There are several complex features such as LSA score and HITS score, which are also used in CRF [21].

3.1.4. Query based extractive approach

In query based text summarization [22], the sentences in a given document are ranked based on the number of occurrence of terms (words or phrases). The sentences containing the query words are given higher ranks than the one which contain a single query word. The sentences with highest rank are extracted into the resultant summary with their structural context. Important sentences may be extracted from different paragraph or subparagraphs. The summary is the collection of extracted sentences. The drawback of this approach is that it also extracts some of the headings with the sentences [2].

In Query-specific Opinion Summarization (QOS) system [22], input is an opinion question. The system gives the summary which is related with the opinion and target described by the question. The system has several steps [23]: a question analysis and query reformulation, a latent semantic indexing based sentence scoring, sentence polarity detection, and a redundancy removal.

3.1.5. Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency [2] is a numerical statistic approach in which it reflects the importance of a word for a document in a collection or corpus. It is often used for weighting factor in information retrieval and text mining. The TF-IDF value increases with respect to the number of times a word appears in the document. It helps in identification of some words, which are generally more common than others. TF computes the frequency of the term in the provided document and it can be computed by Equation (1). IDF computes the importance of the term in a document, using Equation (2). The TF-IDF value can be calculated from TF and IDF values of the term using Equation (3). TF-IDF is mostly used to solve multi-document summarization problem.

$$TF(t) = \frac{\text{Number of times term t occures in document}}{\text{Total Number of terms in the document}} \quad (1)$$

$$IDF(t) = \log(\frac{\text{Totla Number of Documetns}}{\text{Number of documents containg term } t}) \quad (2)$$

$$TF - IDF(t) = TF(t) \times IDF(t) \quad (3)$$

3.1.6. Latent semantic analysis (LSA)

LSA is an automatic statistical and mathematical technique. It is used for extracting and concluding relations of expected contextual usage of words in passages of discourse. It is not a traditional NLP or AI program. It does not use humanly constructed dictionaries, knowledge bases, semantic networks, and syntactic parsers. It takes parsed words as input which are defined as unique character strings.

The first step in LSA is to generate the term-document matrix [24], which contains the rows as the unique words of documents and the columns as the documents or the paragraphs of a given document. The next step is to generate the document-document or paragraph-paragraph matrix and term-term matrix using the first term-document matrix. The generated two matrices, doc-doc matrix and term-term matrix will help in computation of the Singular Value Decomposition (SVD) matrix. SVD identifies the patterns and relationship between the terms and the provided text. The comparison of words is computed by considering the cosine value of the two vectors generated by their respective two rows. Words are related if the value is close to 1, and 0 indicates unrelated words.

LSA gets its name because of SVD, which is applied to document-word matrices; document or text groups that are semantically related to each other, even if there are no common words. Words that have same style are also in the same singular space. LSA can be applied to extract the topic-words and sentences from documents.

3.1.7. Lexical chain analysis for summarization

A lexical chain is a sequence of related words in semantic manner. The chain is autonomous of all the grammatical structure of the text. It is a candidate set that acquires cohesive structure of the text. A lexical chain provides the meaning that the term represents, and also provides the understanding of an unclear term. There are several lexical chain algorithms developed and used by researchers, which are listed in Table 2. The focus of these algorithms is on the selection of candidate set. Candidate set is nothing but a set of keywords which are extracted from document.

Lexical Chain is a most effective technique for identifying semantically related concepts in text [25]. The simplest form of cohesion, relatedness between elements, is lexical cohesion. Lexical chains represent the lexical cohesion among an arbitrary number of related words. Lexical chains can be found by recognizing groups of words that are semantically related to each other. Using lexical chains in text summarization is efficient because these relations are easily identifiable within the source text, and vast knowledge bases are not necessary for computation. We can statistically compute the most important information by looking at structure of the document rather than deep semantic meaning. Lexical chain can be calculated using knowledge databases or lexical corpus, which contains terms with their proper POS tags and their associations [26]-[30].

3.2. Features for Extractive Summarization

For selecting appropriate sentences in to generate summary, some features have to be considered. Features useful for extractive based text summarization [25] are listed here. The provided features are surveyed on English language text documents.

3.2.1. Sentence location feature

Generally, the first and the last sentences of the first and the last paragraph of any text document have more information. Usually the first sentence provides the information about main topic and the last sentence provide the conclusion of that paragraph. Such sentences have higher chances of getting included

in summary.

3.2.2. Sentence length feature

Too small and too large sentences are usually not included in summary. This feature can be found by just finding the sentence length and putting threshold value for lower and upper bound. If the summary length is enough i.e. more than 50 percent, then this feature can be avoided.

3.2.3. Proper noun feature

Sentences which contain the proper nouns are having greater chances for including in summary. Proper noun can be a set of collections like a person, place, city, organization, etc. This feature can be extracted by using Name Entity Recognition (NER) of Natural Language Processing (NLP).

3.2.4. Pronouns

Pronouns such as "it, they, etc." should not be in summary or can be replaced by its corresponding nouns. If proper noun can be identified, the pronouns can be replaced by its corresponding proper noun.

3.2.5. Cue-phrase feature

Sentences which contain cue-phrase are most likely to be considered in the summary. Cue-phrase words are like develop, summary, conclude, results, etc. To provide this type of feature there is at least one corpus which contains the same cue-phrase for comparison.

3.2.6. Font based feature

Sentences which are in bold, italic, or underlined have higher chance for sentence extraction. However, in many documents the words with this format are headings or notations for additional information.

3.2.7. Occurrence of non-essential information

Some words such as additionally, furthermore, etc. provide non-essential information. These words are generally occurring at the beginning of the sentence, so it is easy not to consider that sentence in summary.

These features provide linguistic and statistical information of text. There are numbers of approaches, which use these important features for summarization. We have decided to use sentence length, sentence position, pronoun, and proper noun features for extractive based text summarization using lexical chain analysis.

3.3. Problems with Extractive Summarization

Gupta *et al.* in 2010 [2] presented some problems in extractive based text summarization. Usually extracted sentences have much higher length with respect to information it acquires. Due to this reason, segments which are unnecessary for summary also occupy space. Relevant information is normally spread across many sentences; which extractive summary do not carry in summary. The problem can be solved, if summary length is enough to hold all necessary sentences. Moreover, an extractive summary cannot present the conflicting information accurately.

4. Evaluation Measures and Tools for Automatic Summarization

In this section, we provide the evaluation measures for automatic text summarization. Section 4.1 provides the evaluation measures for automatic text summarization and Section 4.2 provides different evaluation tools available for automatic text summarization.

4.1. Evaluation Measures of Automatic Text Summarization

The evaluation of system generated summary has been performed by comparing the generated summary with the ideal summary, which is generally prepared by an expert person of the domain. There are two types of evaluation methods used by researchers: Intrinsic and Extrinsic. In Intrinsic (or normative) evaluation, the generated summary can be compared with standard summary (human effort). Extrinsic

evaluation judges quality of generated summary according to how it will effect on the completion of some other task. For this research work, we have used intrinsic evaluation method. There are two important measures for evaluating system generated summary, which must be measured at the time of intrinsic evaluation [31] [8].

1) Compression Ratio: It indicates "how much shorter the summary is"? It can be computed using Equation (4).

$$\text{Compression Ratio (CR)} = \frac{\text{length of Summary}}{\text{length of Full Text}} \quad (4)$$

2) Retention Ratio: How much information is acquired in the summary? It can be computed using Equation (5).

$$\text{Retention Ratio (RR)} = \frac{\text{information in Summary}}{\text{information in Full Text}} \quad (5)$$

The researchers use human generated summary as an ideal summary due to lack of availability of any standard ideal summary. Human generated summary is very useful technique as it allows use of multiple reference summaries rather than using a single ideal summary. There are many tools developed for this purpose [32], which are discussed next. All tools required reference summary as the ideal summary. However, the best way to select reference summary is to use a human generated summary.

4.2. Tools Available for Evaluation of Automatic Text Summarization

After performing summary extraction of the given document, the next step is evaluation of the generated summary. In this section, we present some important evaluation techniques.

4.2.1. Summary evaluation environment (SEE)

SEE [32] provides interface in two separate panels, which provides the evaluation result in a separate file. SEE supports both extrinsic and intrinsic evaluation approaches.

4.2.2. MEADeval

MEADeval [32] is a Perl toolkit for multi-lingual summarization and evaluation and Document Understanding Conference (DUC)-style extracts evaluation. It supports many standard metrics (measures) such as precision, recall, kappa, and relative utility. MEADeval compares the system summary to a reference summary (or "ideal" summary).

4.2.3. Recall-oriented understudy for gisting evaluation (ROUGE)

ROUGE is a tool and set of metrics used for evaluation of automatic summarization in natural language processing. ROUGE [32] uses unigram co-occurrences between summary pairs. The evaluation procedure provides well results with human evaluation. ROUGE is recall oriented and separately provides an evaluation for 1, 2, 3 and 4 -grams. ROUGE has been tested for extraction based text summarization with content overlap [33]. ROUGE provides following different metrics.

- ROUGE-S: Skip Bigram Statistics
- ROUGE-N: ROUGE n-gram Statistics
- ROUGE-SU: Advance of ROUGE-S (With unigram)
- ROUGE-L: Longest Common Subsequence (LCS) Based Statistics
- ROUGE-W: Weighted LCS based Statistics

We have used ROUGE to evaluate system generated summary, we have considered a human generated summary as an ideal summary and taken it as a reference. We have used ROUGE-S, ROUGE-SU, and ROUGE-N metrics for evaluation purpose, which is listed in Table 3. The ROUGE provides the ability to evaluate summary using multiple ideal references.

5. Proposed Approach: Text Summarization Using Lexical Chain Analysis

In this section, we present our proposed approach which uses lexical chain analysis for generating text summarization. Section 5.1 provides a survey on lexical chain generation approaches. Section 5.2 provides the proposed lexical chain generation approach with sample result of generated lexical chains. It also provides the evaluation results of generated summary.

5.1. Survey on Lexical Chain Generation Approaches

For English language words, WordNet [26] is often used as the lexical resource for identifying term relatedness, which provides the relationship types such as synonymy, hypernym, and hyponymy. The first lexical chain for summarization is introduced by Morris and Hirst [27] in 1991. They described the steps, but did not implemented because of the lack of availability of the electronic corpus or any lexical resource. The lexical corpus provides the functionality to find the relation between terms, such as synonymy, siblings, hypernyms, and hyponyms. Moreover, Barzilay and Elhadad [28] showed that implementation of the theoretical work proposed by Morris and Hirst [27] could be possible for automatic document summarization. However, the algorithms take exponential time to generate lexical chains. Later on, a linear time algorithm was proposed and implemented in 2002 by Silber and Mccoy [25]. All of the mentioned work uses WordNet as a lexical database for English text summarization.

For English text summarization, Roget's Thesaurus was suggested by Mario *et al.* in 2003 [34], which is used for lexical chain generation. Mario *et al.* [34] presented two relations to find word sense disambiguation. The first one is the repetition of same word and the second one is the inclusion of the same head-word using Roget's Thesaurus. The second step of lexical chain generation is to insert the word in the chain via thesaural relation. Hirst and St-Onge in 1995 [29] provided one more linguistic task with lexical chains, the detection and correction of malapropisms. A malapropism is an intended word, which can be replaced with another word of similar sound or similar spelling but has quite different meaning e.g., an ingenuous (original is ingenious) machine for peeling oranges.

Table 2 provides the different lexical chain generation approaches used by researchers and the results obtained by them in the form of precision and recall with time complexity. Table 2 also provides the different word sense used by different researchers. The word sense can be computed by using different corpora or knowledge sources.

Table 2. Different Lexical Chain Generation Approaches

Algorithm	Author	Candidate Set	Time Complexity	Word Sense Disambiguation	Results
Lexical Chain from Corpus	Hirst and St-Onge [29]	All words except stop-words	--	WordNet as a knowledge source	40% words of corpus generates chain
Linear Lexical Chain Algorithm	Silber, H. Gregory, Kathleen F. Mccoy [25]	All Nouns	O(n)	Combination of Synonyms, Hypernyms and Hyponyms using WordNet	85% Precision
Dynamic Chaining Algorithm	Barzilay and Elhadad [28]	All Nouns & Nouns Compounds	$O(n^2)$	WordNet and systematic relations.	61% Precision 67% Recall
Head Words Lexical Chain Algorithm	Jarmasz and Szpakowicz [34]		$O(n^2)$	Repetition of Same Word and Head Roget's Thesaurus	--
Coreference Chain Algorithm	Azzam Saliha, Kevin Humphreys and Robert Gaizauskas [14]	Set of Nouns with Co-Reference chain	--	Considering 'best parse' of two sentences	67% Precision 33% Recall

The researchers used different candidate sets to generate lexical chain, which is listed in Table 2. There are two governing criteria for evaluation of lexical chain: chain strength and chain quality, Morris at al. in 1991 [27] identified following three factors for evaluating chain strength: reiteration, density, and length of

the chain. The higher the factor value, the stronger it is. There is no objective evaluation for the quality of the chain. Another way to evaluate lexical chain is evaluation of text summarization as suggested by Meru *et al*. in 2001 [30]. The ordering of the lexical chain can be done by three ways: by its length, by the number of terms it contains and by the density of the chain.

5.1.1. Proposed approach: Text summarization using lexical chain

In this section, we proposed lexical chain approach for text summarization. We have performed lexical chain analysis using linguistic pre-processing. The linguistic pre-processing includes sentence segmentation, tokenization, POS tagging, entity detection, relation detection respectively in order. We have extracted candidate set (nouns and nouns compounds) in linguistic pre-processing step from the given text document. The lexical chain has been generated using the extracted candidate set. We have used WordNet lexical corpus to check word sense. The sample output of generated lexical chain is given below. The chain is generated by taking sample text on Data Mining from [2]. The proposed algorithm of lexical chain generation is as follow:

```
For each word in the candidate set {
    For each chain in lexical chain {
        Find Word sense between two words
    }
    If (distance > Threshold)
        Add word to that chain
    Else
        Generate new chain.
}
```

Table 3 provides the sample of generated lexical chains. The result of the generated lexical chain is not promising as it looks. Some chains, highlighted in Table 3 have single or double term which doesn't help in relative information extraction. To improve the strength of the lexical chain we are planning to extend it with co-reference words. We are also going to apply new methods to find word sense between two words.

Table 3. System Generated Sample Output of Lexical Chains

Sr. No.	Lexical Chain
1.	Quantity, Attributes, Time, Number, Amount, Portion, Amount, Example, Date, Length, Components, Model, Amount, Fraction
2.	**Repository**
3.	**Gap**
4.	Transition, Text, Articles, Books, Text, Publications, Text, Text, Publication, Text, Text, Text, Text
5.	Picture, Document, Documents, Document, Title, Documents, Documents, Documents, Documents, Documents, Documents
6.	**Analysis, Analysis, Analysis, Research**
7.	**Computer, Web, Web**
8.	Science, Engineering, Studies, Focus, Libraries, Studies, Importance
9.	Techniques, Methodology, Techniques, Techniques, Techniques
10.	Demands, Classification, Collections, Sources, Authors, Indexing

Table 4. System Generated Vs Model Summary

System Generated Summary	Model Summary created by Domain Expert
Nowadays, large quantity of data is being accumulated in the data repository. This transition won't occur automatically, that's where Data Mining comes into picture. In Exploratory Data Analysis, some initial knowledge is known about the data, but Data Mining could help in a more in-depth knowledge about the data. Seeking knowledge from massive data is one of the most desired attributes of	In today's era, quantity of data is being accumulated in the data repository in exponential rate. In Data Analysis, some initial knowledge is known about the data, but Data Mining could help in a more in-depth knowledge about the data. Seeking knowledge from massive data is one of the most desired attributes of Data Mining. A number of Data Mining techniques are developed to mine this vast amount of data.

Data Mining. Manual data analysis has been around for some time now, but it creates a bottleneck for large data analysis. Fast developing computer science and engineering techniques and methodology generates new demands to mine complex data types. A number of Data Mining techniques (such as association, clustering, classification) are developed to mine this vast amount of data. Previous studies on Data Mining focus on structured data, such as relational and transactional data. However, in reality, a substantial portion of the available information is stored in text databases (or document databases), which consists of large collections of documents from various sources, such as news articles, books, digital libraries and Web pages. Text databases are rapidly growing due to the increasing amount of information available in electronic forms, such as electronic publications, e-mail, CD-ROMs, and the World Wide Web (which can also be viewed as a huge, interconnected, dynamic text database). Data stored in text databases is mostly semi-structured, i.e., it is neither completely unstructured nor completely structured.

Text databases are rapidly growing due to the increasing amount of information available in electronic forms, such as electronic publications, e-mail, CD-ROMs, and the World Wide Web.

Data stored in text databases is mostly semi-structured, i.e., it is neither completely unstructured nor completely structured. In recent database research, studies have been done to model and implement semi- structured data. Information Retrieval techniques, such as text indexing, have been developed to handle the unstructured documents. But, traditional Information Retrieval techniques become inadequate for the increasingly vast amount of text data. Users need tools to compare different documents, rank the importance and relevance of the documents, or find patterns and trends across multiple documents. Thus, Text Mining has become an increasingly popular and essential theme in Data Mining.

Table 5. Evaluation Results of Summarization System using Proposed Lexical Chain Analysis Algorithm

Summary Models	Lexical Chain Generated by Unique Terms									Lexical Chain Generated by Chain Length								
	ROUGE_N			ROUGE_S			ROUGE_SU			ROUGE_N			ROUGE_S			ROUGE_SU		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Model1	0.58	0.49	0.53	0.53	0.53	0.53	0.29	0.26	0.27	0.59	0.52	0.55	0.55	0.55	0.55	0.29	0.27	0.28
Model2	0.60	0.34	0.44	0.40	0.40	0.40	0.25	0.20	0.22	0.54	0.32	0.40	0.37	0.37	0.37	0.23	0.18	0.20
Model3	0.71	0.53	0.61	0.57	0.57	0.57	0.32	0.28	0.30	0.58	0.44	0.50	0.49	0.49	0.49	0.28	0.24	0.26
Model4	0.71	0.53	0.61	0.57	0.57	0.57	0.32	0.28	0.30	0.58	0.44	0.50	0.49	0.49	0.49	0.28	0.24	0.26
Model5	0.55	0.57	0.56	0.58	0.58	0.58	0.29	0.29	0.29	0.50	0.53	0.51	0.55	0.55	0.55	0.27	0.27	0.27
Model6	0.56	0.44	0.49	0.49	0.49	0.49	0.27	0.24	0.26	0.56	0.46	0.51	0.51	0.51	0.51	0.28	0.25	0.26
Average	0.62	0.48	0.54	0.52	0.52	0.52	0.29	0.26	0.27	0.56	0.45	0.50	0.49	0.49	0.49	0.27	0.25	0.26

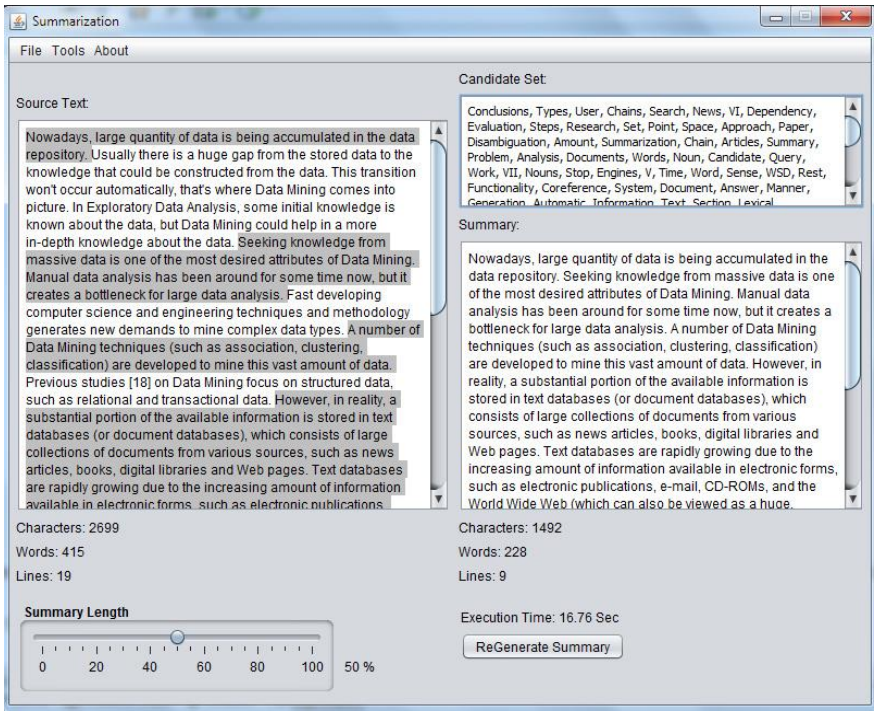

Fig. 2. GUI of automatic text summarization system using lexical chain analysis.

Table 4 provides the result of system generated summary and model summary created by a domain expert. Red text form system generated summary indicates matching content with Model Summary. Blue text from Model Summary indicates the additional text which was not considered by the System Generated Summary. Table 5 provides the evaluation results with two lexical chain scoring factors, chain length and chain unique terms. The Table 5 shows the result using three metrics, ROUGE_N, ROUGE_S and ROUGE_SU. For the evaluation purpose, we have selected a sample text from document [2]. The result is generated by comparing system generated summary with human (expert) generated summary. Model1 to Model6 provides the human generated summary. Fig. 2 presents developed system's GUI for the automatic text summarization using lexical chain analysis.

6. Conclusion

In this paper, we have outlined various automatic text summarization techniques based on an extractive based approach. We presented linguistic pre-processing steps along with available tools and plugins. We also explained the extractive based text summarization process. We have outlined some of English text features, useful in extractive based text summarization. We proposed lexical chain approach for automatic text summarization and we generated lexical chains using WordNet lexical library. We also provided the evaluation results of our system generated summary with human generated summary. Our future plan is to improve the strength of lexical chain by improving the word sense measurement and to consider co-reference chains along with the lexical chain for automatic text summarization.

References

[1] Nenkova, A. (2011). Automatic summarization. *Foundations and Trends® in Information Retrieval*, *5(2)*, 103–233,.

[2] Gupta, V., & Lehal, G. S. (2010). A survey of text summarization extractive techniques. *Journal of Emerging Technologies in Web Intelligence, 2(3)*, 258–268.

[3] Goldstein, J., Carbonell, J., & Kantrowitzt, M. (1998). *Multi-Document Summarization By Sentence Extraction*, 40–48.

[4] Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM*, *16(2),* 264–285.

[5] Kupiec, J., Pedersen, J., & Chen, F. (1995). A trainable document summarizer. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development In Information Retrieval.*

[6] Klein, D., *et al*. (2003). Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (pp. 423–430).

[7] Prajapati, H. B., & Dabhi, V. K. (2010). XML based architectures for documents comparison, categorisation, and scrutinisation. *International Journal of Data Analysis Techniques and Strategies, 2(4),* 385–410.

[8] Goldstein, J., *et al*. (1999). Summarizing text documents: Sentence selection and evaluation metrics. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[9] Zajic, D. M., Bonnie, J. D., & Jimmy, L. (2008). Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management, 44(4),* 1600-1610.

[10] Ferreira, R., *et al.* (2013). Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications, 40(14)*, 5755-5764.

[11] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

[12] Fouziasulthana, K., & Kanya, N. (2013). Content extraction with text mining using natural language

processing for anatomy based topic summarization. *3(1)*, 564–567.
[13] Maybury, M. T., & Andrew, E. M. Jr. (Nov. 2005). Automated segmentation, information extraction, summarization, and presentation of broadcast news. U.S. Patent No. 6,961,954.
[14] Azzam, S., Humphreys, K., & Gaizauskas, R. (1999). Using coreference chains for text summarization. *Proceedings of the Workshop on Coreference and Its Applications. Association for Computational Linguistics* (pp. 77–84).
[15] Stanford CoreNLP. From http://nlp.stanford.edu/software/corenlp.shtml
[16] Apache OpenNLP. From http://opennlp.apache.org/
[17] Natural Language Toolkit. From http://nltk.org/
[18] Rapid Miner. From Available: http://rapidminer.com/
[19] Khosrow, K. (June 2004). *Automatic Text Summarization with Neural Networks*.
[20] Fattah, M. A., & Ren, F. Automatic text summarization. *International Journal of Computer Science*, *3(1)*, 25–28.
[21] Shen, D., Sun, J., Li, H., Yang, Q., & Chen, Z. (2004). *Document Summarization Using Conditional Random Fields*, 2862–2867.
[22] Pe, F. C., *et al*. Automated querybiased and Structure preserving text summarization on web documents. *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications*. İstanbul.
[23] Varadarajan, R., & Vagelis, H. (2005). Structure-based query-specific document summarization. *Proceedings of CIKM'05*, ACM, Bremen, Germany.
[24] Gong, Y. (2001). *Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis*.
[25] Silber, H. G., & Mccoy, F. K. (2002). *Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization*.
[26] Erekhinskaya, T., & Moldovan, D. (2013). *Lexical Chains on WordNet and Extensions*, 52–57.
[27] Morris, H. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text.
[28] Barzilay, R., & Elhadad, M. (1997). Using lexical chains for text summarization. *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, *17(1)*.
[29] Hirst, G., & St-onge, D. (April 1995). Lexical chains as representations of context for the detection and correction of malapropisms.
[30] Brunn, M., Chali, Y., & Pinchak, J. C. (2001). Text summarization using lexical chains. *Proceedings of Workshop on Text Summarization in Conjunction with the ACM SIGIR Conference*.
[31] Nenkova, A., & Kathleen, M. (2012). A survey of text summarization techniques. *Mining Text Data*. Springer US. 43-76.
[32] Thesis, L. (2004). Evaluation of automatic text summarization. *KTH Numerical Analysis and Computer Science*.
[33] Liu, F., & Liu, Y. (Jan. 2010). Exploring correlation between ROUGE and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech, and Language Processing*, *18(1)*, 187–196.
[34] Jarmasz, M., & Szpakowicz, S. (2003). Not as easy as it seems : Automating the construction of lexical chains using roget's thesaurus. *Advances in Artificial Intelligence*, Springer Berlin Heidelberg. 544–549.

Sagar M. Patel is currently working as an Assistant Professor at Information Technology Department, Charusat, Anand, Gujarat, India. His research areas are web technologies, artificial intelligence, text mining, evolutionary computing. He had completed the M. Tech. in information technology from DDIT, DDU, Nadiad in 2014. He had done his dissertation on

"Extractive based Automatic Summarization". He received his bachelor's degree in information technology from CCET, Wadhwan, Saurastra University Rajkot in 2010.

Vipul Dabhi is currently working as an Associate Professor at Information Technology Department, Dharmsinh Desai University, Nadiad, Gujarat, India. His research areas are evolutionary computing, time series modeling, distributed computing, and service oriented computing. He has published more than 30 research papers in International / National Conferences and Journals.

He completed Ph.D. in computer engineering from Dharmsinh Desai University in 2014. His Ph.D. dissertation title was "Time Series Modeling and Prediction using Postfix Genetic Programming". He received the master of engineering (M.E.) in computer engineering from Dharmsinh Desai University in 2006. He did his dissertation on "Satellite Based Grid Computing" at Space Application Center (SAC), ISRO, Ahmedabad. He received the bachelor of engineering (B.E.) in instrumentation and control from Gujarat University in 2000.

Harshadkumar B. Prajapati is an Associate Professor at the Department of Information Technology, Faculty of Technology, Dharmsinh Desai University, India. He completed the PhD in computer engineering from Dharmsinh Desai University in 2014. He received ME in computer engineering from Dharmsinh Desai University in 2007 and a BE in electronics and communication from Gujarat University, India, in 2000. He has published several papers in international journals and conferences. He has served as a reviewer in international journals and conferences. His research areas include distributed computing, grid computing, workflow scheduling and cloud computing. Currently his research focuses on cloud computing, bigdata analytics, and machine learning.