

# **Intelligent Summarization: Leveraging Cohesion in Text**

**Arun Thundyill Saseendran**

## **A Dissertation**

Presented to the University of Dublin, Trinity College  
in partial fulfilment of the requirements for the degree of

**Master of Science in Computer Science (Future Networked  
Systems)**

Supervisor: Professor Khurshid Ahmad

August 2019

# Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

---

Arun Thundyill Saseendran

August 14, 2019

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

---

Arun Thundyill Saseendran

August 14, 2019

To my mother, Anitha S and my father, Saseendran TV for what I am and for what I will be! To my teachers for moulding me! To the Almighty God!

For Good Health, Joy and Prosperity!

# Acknowledgments

My sincere thanks to Professor Khurshid Ahmad for his supervision and guidance throughout the work. I am grateful to him for patiently explaining the concepts and giving me motivation throughout.

My special thanks to my mother Anitha S, my father Saseendran TV, my sweet sister Deepa TS and her little one Aarav for providing me moral support from over a thousand miles away.

I would like to thank Dr. Husanbir Singh Pannu, Post-Doctoral researcher, Trinity College Dublin, and Shane Finan, Research Manager, FIOSIN for always willing to help, giving me support, and encouragement.

Thanks to my friend and peer Viren Chhabria for being a good friend and for the motivation. I would also like to thank all my friends, near and dear for understanding my mood - good and bad throughout the course of this work, and still supporting me.

ARUN THUNDYILL SASEENDRAN

*University of Dublin, Trinity College  
August 2019*

# Intelligent Summarization: Leveraging Cohesion in Text

Arun Thundyill Saseendran , Master of Science in Computer Science  
University of Dublin, Trinity College, 2019

Supervisor: Professor Khurshid Ahmad

Text summarization is a task that requires the application of human intelligence in which the human shows an understanding of natural language and can process language, where human beings show creativity by presenting complex objects and events. Human beings use a collection of words in discussing specific subjects - the so-called specialist or scientific lexicon, that comprises the ontology of a domain. A computer program that can mimic these aspects of human intelligence is referred to as an information extraction system, and text summarization systems of the type discussed in this thesis are called extractive text summarization systems. In this work, a text summarization system is presented that is based on the theory of lexical cohesion, where the focus is on how a (scientist) writer repeats a specific word to convince his or her reader about the importance of the theme. The *Intelligent Text Summarization* algorithm makes use of repetition that can be visualized as a graph where the links between nodes represent the linked sentences based on the same word or its close variants. The novelty introduced is the autonomous selection of domain-specific words to produce a readable summary selecting pre-existing sentences in the text. The average summary is about 25% of the text. The system (developed using Java and Python) was tested using computer science texts and was tested using texts in bio-medicine, specifically gastroenterology. An expert compared the summaries of 15 papers generated by the application and found that in 73.6% of the cases the summary was good.

# Summary

Specific language text which is generally non-narrative in nature unlike literary texts like fiction. These texts are a source of valuable information and knowledge which needs to be effectively summarized to be useful. The summaries may be used for a variety of purposes: for understanding texts; for storing and subsequently retrieving text from databases; for indexing; for author attribution; in search engines etc.

Summarization or precis writing is an ancient art of creating abstracts [1]. However, abstract creation is a human-intensive process and requires subject matter experts of the domain to create meaningful summaries. Summarization requires the understanding of natural language text and human intelligence to identify the important part of a text to create a summary. Automatic text summarization is an open research topic since the 1950s and attributes to the field of *Information Science* since it requires the computer to mimic human intelligence in order to create a readable meaningful summary. Creation of a meaningful summary without the loss of important information in a text without human intervention is termed as *Automatic Text Summarization*. The process of automatic summarization can be broadly classified into abstractive summarization and extractive summarization. In abstractive summarization, the summary is created by extracting the key ideas and modifying the sentences of the text. Whereas, in extractive summarization, the key sentences of a text are selected and reordered without any modification to the original sentences to create the summary. In this work, *extractive summarization* is attempted.

The focus of this work is on automatic text summarization of non-narrative scientific text. The choice of the specific domain of scientific texts is made on the understanding that the authors of the scientific texts follow a theme in their writing and make use of domain-specific terms to introduce, define and elaborate new topics resulting in a repetition of important terms throughout the document. These terms can be identified and counted using simple mechanical means to establish a relationship between sentences and to identify the most important sentence to create an extractive summary.

Several approaches have been suggested for extractive summarization in the past works based on various approaches including machine learning methods which require huge training dataset and the model created is only as good as the data it is trained on.

In this work, the algorithm for automatic text summarization of text is devised on the hypothesis that 'scientific messaging relies on the repetition of key terms'. Using this hypothesis an algorithm has been devised that can summarize any arbitrary scientific text by intelligently identifying the key terms in the text, use the key terms to establish relationship between sentences, identify the most important sentences based on the relationships and use them for summarization. The work presented in this thesis is an extension of the work by Benbrahim and Ahmad [2]. Figure 1 shows an easy to understand illustrative flow of the *intelligent text summarization* algorithm designed and implemented (developed using Java and Python) in this work. The figure shows the step by step approach of how an output summary is generated from an input text. This work is planned to be published as a journal article and the work is in progress [3].

The algorithm devised is implemented as a working software to test its practicability. The software developed is also instrumented to perform various systematic, controlled statistically verifiable automatic evaluation of the working of the algorithm. Further, texts from other domains are summarized using the software and subjectively evaluated



by subject matter experts of the domain to verify the working of the software and in turn the algorithm across different domains of scientific text.

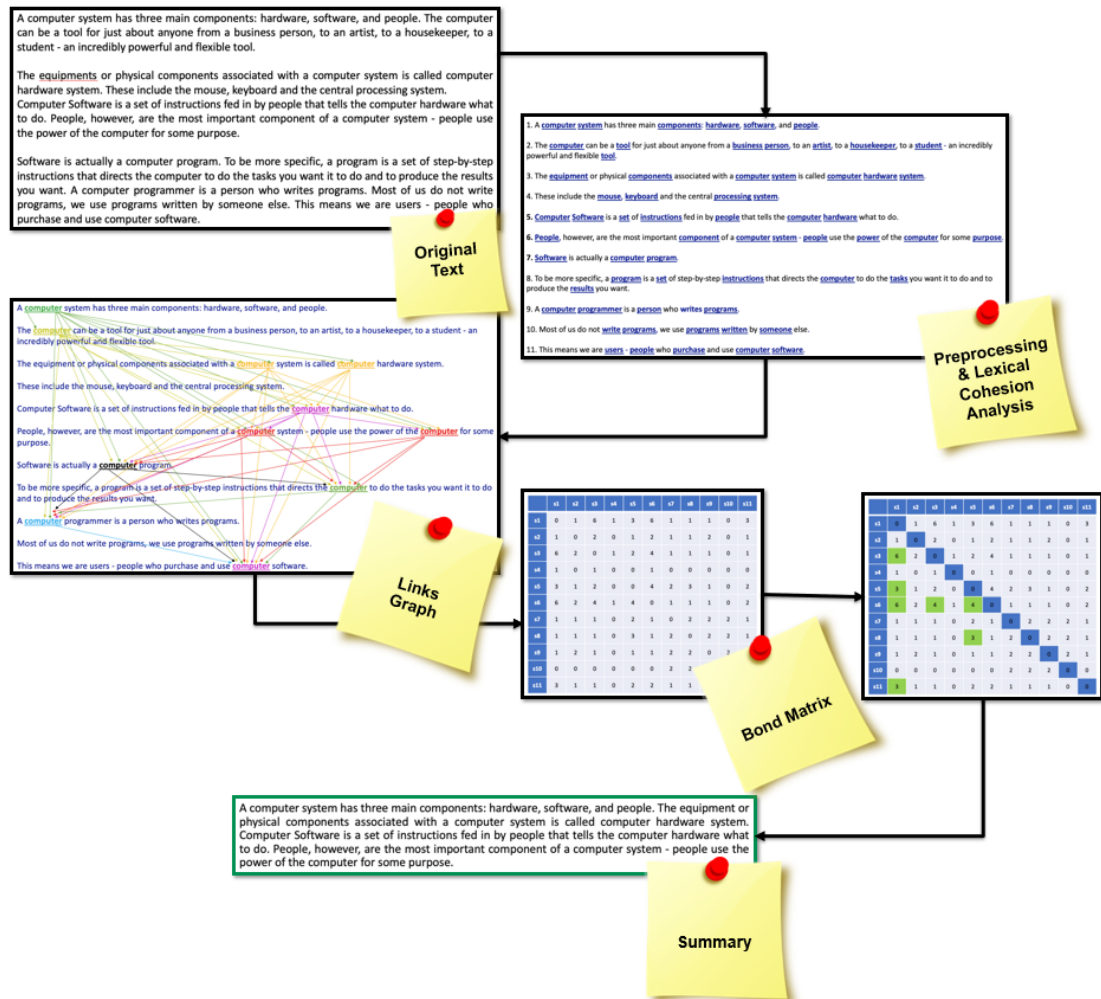


Figure 1: Illustrative flow of the Intelligent Text Summarization Algorithm

# Contents

<b>Acknowledgments</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Summary</b>	<b>vi</b>
<b>List of Tables</b>	<b>xii</b>
<b>List of Figures</b>	<b>xiii</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Problem Definition . . . . .	4
1.2 Contributions . . . . .	7
1.3 Structure of the Dissertation . . . . .	9
<b>Chapter 2 Motivation and Literature Review</b>	<b>11</b>
2.1 Motivation . . . . .	11
2.2 Literature Review . . . . .	13
2.2.1 Lexical Cohesion . . . . .	15
2.2.2 Syntactic Cohesion . . . . .	16
2.2.3 Links and Bonds . . . . .	16
2.2.4 Soft-computing and fuzzy systems . . . . .	17
2.3 Conclusion . . . . .	17
<b>Chapter 3 Method</b>	<b>20</b>
3.1 Introduction . . . . .	20

3.2	Pre-Processing the Text . . . . .	24
3.3	Intelligent Text Summarization Algorithm . . . . .	24
3.3.1	Key Terms Identification . . . . .	27
3.3.2	Sentence Linking and Link Matrix . . . . .	44
3.3.3	Sentence Bonding and Bond Matrix . . . . .	47
3.3.4	Sentence Categorization using in-out bond ratio . . . . .	49
3.3.5	Summary Generation . . . . .	51
3.4	Evaluation Method . . . . .	53
3.4.1	Readability Index . . . . .	53
3.4.2	Correlation of Relative Frequency . . . . .	55
3.4.3	Cumulative Relative Frequency Comparison of top open-class words . . . . .	57
3.5	Conclusion . . . . .	58
<b>Chapter 4 Implementation and Case Studies</b>		<b>61</b>
4.1	Introduction . . . . .	61
4.2	Dataset Selection . . . . .	61
4.3	Technical Specification and System Design . . . . .	63
4.3.1	System Architecture . . . . .	63
4.3.2	Technical Stack . . . . .	66
4.3.3	User Interface Design . . . . .	66
4.4	Case Studies and Discussion . . . . .	76
4.4.1	Controlled Evaluation on Computer Science Text . . . . .	76
4.4.2	Controlled Evaluation on the Bio-medicine dataset . . . . .	80
4.4.3	Experimental evaluation of summarization of large block of tweets . . . . .	81
4.5	System Performance . . . . .	82
4.6	Security and Privacy Concerns . . . . .	82
4.6.1	Security Concerns . . . . .	82
4.6.2	Privacy Concerns . . . . .	85
4.7	Summary . . . . .	86
<b>Chapter 5 Conclusion and Future Work</b>		<b>87</b>
5.1	Conclusion . . . . .	87

5.2 Future Work . . . . .	88
<b>Bibliography</b>	<b>90</b>
<b>Appendices</b>	<b>95</b>

# List of Tables

1.1	Various applications of text summarization with technique highlights and references . . . . .	5
2.1	Comparative analysis of automated text summarization techniques with highlights and dataset . . . . .	18
3.1	Part-of-speech tags used by POS tagger to annotate text . . . . .	34
3.2	Table showing the frequency calculation for candidate terms (W) and other open class words (Wi) for 5 words to the left (L1 to L5) and 5 words to the right (R1 to R5) . . . . .	38
3.3	Flesch Kinkaid Reading Ease interpretation reckoner . . . . .	54

# List of Figures

1	Illustrative flow of the Intelligent Text Summarization Algorithm . . .	viii
1.1	Generic flow of Automated Text Summarization process . . . . .	3
1.2	General Classification of Automated Text Summarization . . . . .	4
1.3	The logo for Curukka Text Processing Pipeline . . . . .	8
1.4	Graphical representation of the research design plan . . . . .	10
2.1	Different types of lexical and syntactic cohesion . . . . .	12
2.2	Summary of studied approaches (Green links show the used approaches)	14
2.3	Overview of the literature survey on text summarization techniques and references for various modes of operation . . . . .	15
3.1	The proposed flow of the algorithm with the core methods . . . . .	22
3.2	Sample scientific text as an example . . . . .	23
3.3	Algorithm of the proposed text summarization technique - Intelligent Text Summarization . . . . .	26
3.4	A sample snapshot of the BNC corpus with words represented as TSV .	28
3.5	Graphical representation selection criteria of terms using the Z-Score of the frequency and weirdness index . . . . .	31
3.6	Design Specification of the selection of terms using Weirdness Index represented using UML Flow-chart . . . . .	33
3.7	Design Specification of the selection of key terms including POS Tagging represented using UML Flow-chart . . . . .	37
3.8	Design Specification of the collocate candidate selection represented us- ing UML flowchart . . . . .	43
3.9	Logical view of the sentences annotated with the identified key terms .	44

3.10	Network among various sentences based on key terms for incoming, outgoing and intermediate sentence analysis . . . . .	46
3.11	The link matrix showing the count of links between sentences . . . . .	46
3.12	The Bond matrix showing the detection of bonds from link matrix when the bond strength is set as 2 . . . . .	48
3.13	The Bond matrix showing the detection of bonds from link matrix when the bond strength is set as 2 . . . . .	49
3.14	The summary generated using the Intelligent Summarization Algorithm with the Sample Text as input . . . . .	59
3.15	Design Specification of Intelligent Text Summarization Algorithm UML flowchart . . . . .	60
4.1	System Implementation Design of the Intelligent Text Summarization - Curukka System . . . . .	64
4.2	Screenshot of the configuration panel for the Curukka System . . . . .	68
4.3	Screenshot of the Text Input Screen for the Curukka System . . . . .	69
4.4	Screenshot of the Summary Screen for the Curukka System . . . . .	70
4.5	Screenshot of the Metrics Screen for the Curukka System . . . . .	71
4.6	Screenshot of the Finger Print Analysis Screen for the Curukka System . . . . .	72
4.7	Screenshot of the Signature Analysis for the Curukka System . . . . .	73
4.8	Screenshot of the Collocation Analysis Screen for the Curukka System . . . . .	74
4.9	Screenshot of the File Menu option the Curukka System . . . . .	75
4.10	Screenshot of the Export Menu option in the Curukka System . . . . .	75
4.11	Screenshot of the Export CSV File dialogbox in the Curukka System . . . . .	76
4.12	Comparison of total sentences of the original with the sentences obtained from proposed text summarization method . . . . .	77
4.13	Comparison of total token count of the original with the sentences obtained from proposed text summarization method . . . . .	78
4.14	Comparison of readability index of original with the readability index of the summary text obtained from proposed text summarization method . . . . .	79
4.15	Comparison of cumulative relative frequency of original with the cumulative relative frequency of the summary text obtained from proposed text summarization method . . . . .	80

4.16 Subjective evaluation of the summaries produced for the bio-medicine dataset by the intelligent text summarization algorithm by a subject matter expert on a scale of 10 . . . . .	81
---	----



# Chapter 1

## Introduction

Data is being produced in abundance. In the current era of the information age, the rate at which the data is produced is increasing exponentially day-by-day. At the same time, technology is being used better than ever to harness the knowledge in data. Separate disciplines of engineering such as Knowledge and Data Engineering are gaining traction and active research happens in the area of Information Science to find better methods and techniques to harness the knowledge in data. Of the data being produced, the most prominent form of data is unstructured textual data. Textual data is produced in the form of essays, blogs, microblogs, books, articles, journals, and many other forms. Researchers and industry are working relentlessly to make effective use of these unstructured form of data to provide meaningful information to the public and business alike.

When it comes to textual data, search engines, encyclopedias, research journal databases, etc., are in a need to organize the text effectively to serve its customers. Having access to the well-organized text enables the common public to better make use of the text. One of the important forms of presenting a text for easy understanding and perception is to provide the basic details about the text like the source, author, date, and place of publication, etc., along with the summary of the text. Summaries have proved to be an effective form of representation of the text that is easy for people to understand and perceive. From a kid who wants to know what the story is about to a researcher who wants to know what a research journal is on, relies on the summary.

The art of abstraction or creating summaries is an ancient art and an important

part of Information Science. The summaries for various artifacts were created manually by subject matter experts of the respective domains. In domain specific text summarization, we need a specific domain knowledge. For example, if the user wants to summarize an article in bio-medicine, one needs to have enough domain knowledge to incorporate in the model. In fact, generic summary writing was taught as a subject called 'Precis' writing, the basics of which can be applied by domain experts to summarize texts in their domain. However, with the abundance of text being produced and the text being made more accessible to people, humans cannot produce summaries for all the text. Though most of the scientific text in the recent past come with abstracts as a part of the text written by the authors, the abstracts by authors are not always true summaries of the text. Hence there is a need for automatic text summarization.

## **What is automated text summarization?**

*Automated text summarization* is an automated method which can produce summaries from the text without the loss of any important information.

It is an automatic process to shorten a given text document without loss of information and creating an efficient summary with minimum redundancy. There are few motivations to explore this research area such as (a) Increasing data including velocity, volume and variety (b) Less time to process.

Automatic Text summarization is a very broad area and there are many ways proposed by researchers for text summarization. Figure 1.1 shows a very simple flow for an automated text summarization process. The pre-processing stage is a very generic step that deals with the cleaning, parsing and encoding the input text in a manner that can be used for processing in the next steps. The pre-processing step is not itself a core part of the summarization process, however, it is essential since machine reading of the text in most cases needs processing.

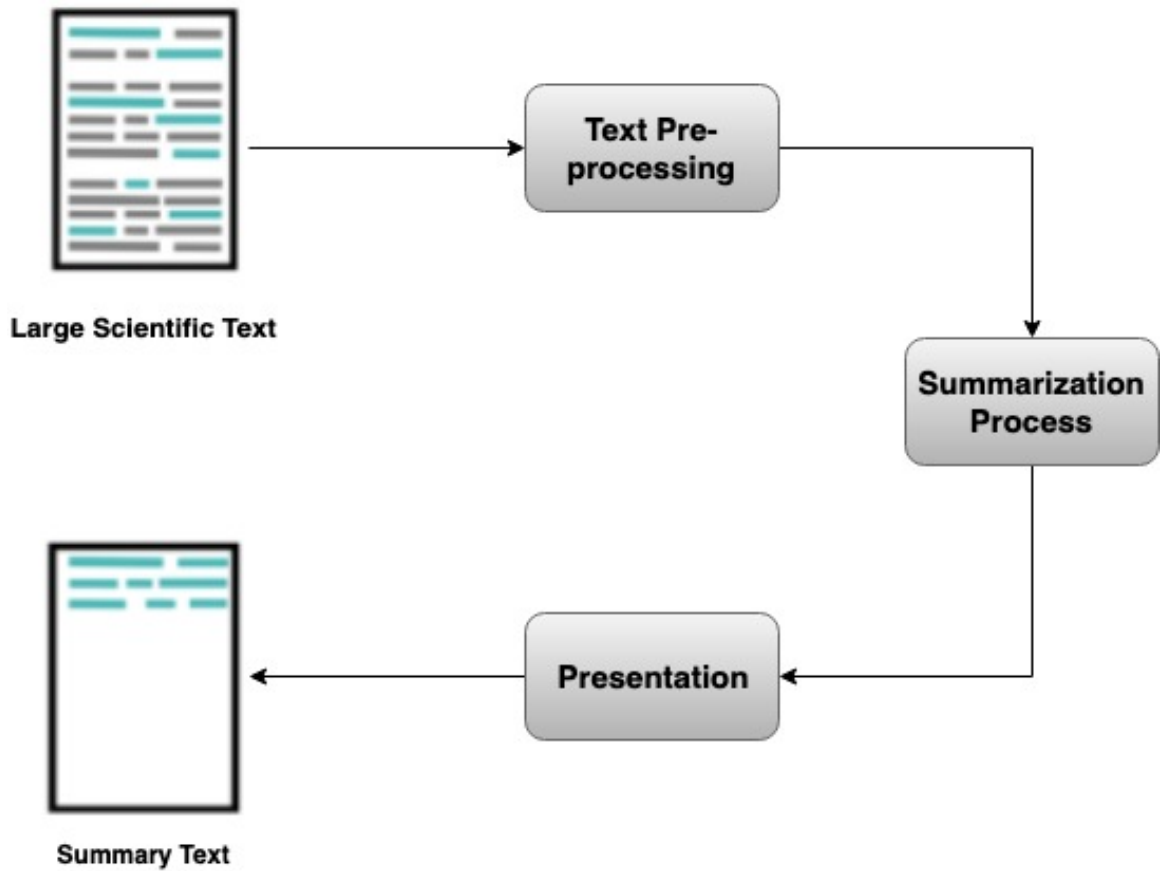


Figure 1.1: Generic flow of Automated Text Summarization process

Once the text is processed and machine-readable, the next step involves the execution of the summarization process. The summarization process can be classified in several ways. The classification is based on either the purpose of summarization, the output type or input type. The generic classification of the summarization is shown figure 1.2.

Based on the purpose it is classified into generic summarization, domain-based summarization which focuses on the creation of summary for a specific domain by leveraging expert domain experience and query-based summarization which focuses on creating a summary based on an input query or keywords. Based on the type of input, summarization can be classified into single-document and multi-document summarization. Single document summarization as the name suggests deals with the summarization of a single document. Whereas in multi-document summarization, it can be the summa-

rization of multiple similar documents or multiple varied documents. Finally, based on the output, summarization is classified as abstractive and extractive. Abstractive summarization is the process in which the summarization method creates the summary by finding the important information in the text and then forms sentences to produce the summary. Extractive summarization is the process in which the sentences representing the most important information is identified and ordered according to the algorithm to produce the summary. The sentences are extracted from the text and not produced by the algorithm, hence, the name extractive summarization.

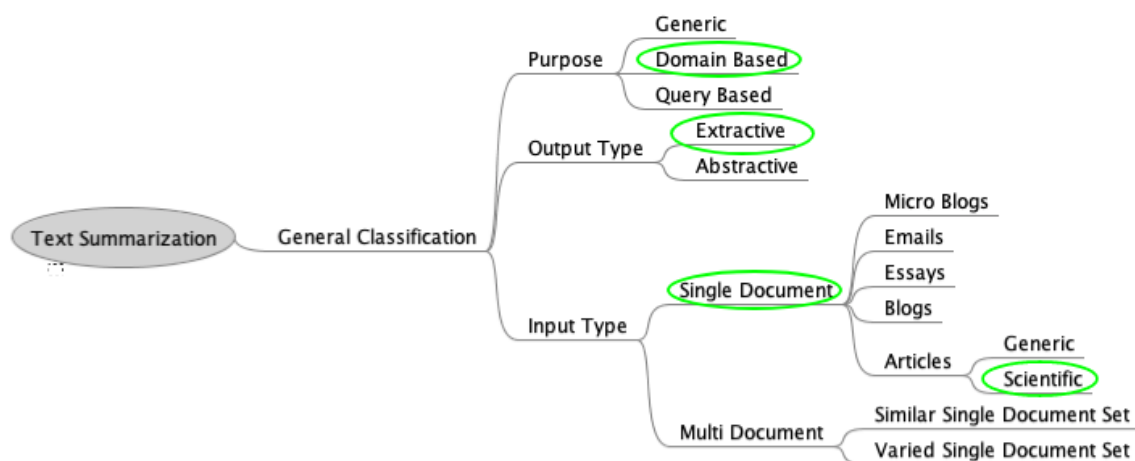


Figure 1.2: General Classification of Automated Text Summarization

## 1.1 Problem Definition

As explained, with all the digitization and with the internet being the endless source of information, textual information is flooding the internet. However, it is really hard to find relevant information in all this amount of text. Hence, automated text summarization is essential. To viewing a snippet of a web-page in a search engine result in viewing the abstract of a research paper in a journal database, summaries play an important role for information dissipation.

The reason to have a text summary is to read quickly and if required the reader can choose to read the entire document, and it would be efficient to have an automated way to create the summary instead of having to go through the detailed text. On the

other hand, that is not enough, and we do need to go into the details. Even in that case, it is relevant to have text summary as to know if the detailed document is useful to read in its entirety. In other words, does this document contain the information, the user is looking for? The research problem to find out text summary automatically are two folds:

1. How to select the most relevant information?
2. How to express the key information in the best possible way?

Thus, the objective is how can we say the most important things in the shortest amount of time and space? We want to optimize the coverage of the topic and optimize its readability. So, the text summary should retain the information content and should be able to present in the best possible readable way which sounds logical and readable.

Some of the applications of text summarization along with the one of the techniques used, the highlights of the work and reference is shown in table 1.1.

Table 1.1: Various applications of text summarization with technique highlights and references

Sr	Application	Technique	Highlights	Ref
1	Automated content creation	Augmented Reality	Augmented reality and text mining to identify virtual contents to match editorial content	[4]
2	Books and literature	Hidden Markov Model	Passage model and token models using HMM	[5]
3	E-learning and class assignments	Summarization Assessment based on Linguistic Knowledge (SALK)	It merges semantic and syntactic information in order to produce an effective evaluation method	[6]
4	Email overload	Deep learning	Unsupervised extractive summarization of emails using a deep auto-encoder with excellent performance	[7]

5	Financial re-search	EA-LTS (extractive and abstractive long text summarization)	In summary generation, attention and pointer mechanism are united with seq2seq	[8]
6	Helping disabled people	Decision tree classifier C5	Sentence segmenter, tokenizer, stopword removal, stemming, tf-idf, sentence similarity to title and decision tree	[9]
7	Internal document workflow Legal contract analysis	LetSum	Documents architecture and its thematic structures to build a table summary for coherency and readability	[10]
8	Medical cases	statistica/graph/tree based	Survey of articles	[11]
9	Meetings and video conferencing Newsletters	feature-based approach	Using prosodic and lexical features maximal marginal relevance, latent semantic analysis to summarization, compared with ROUGE software.	[12]
10	Patent research	text mining	Text segmentation, summary extraction, feature selection, term association, cluster generation, topic identification, and information mapping.	[13]
11	Question answering and bots	Machine learning and NLP	Extractive summarization using sample size, group size, and PICO values from full text PDF reports.	[14]

12	Science and RnD	RnD program planner	Text Mining Text Summarizing Decision Support Knowledge Discovery R&D Planning	[15]
13	Search marketing and SEO	Based on term-frequency and ontology	Pragraph ranking based on relevance between main topics and each individual paragraph	[16]

Though there are several methods already published for text summarization, they rely on machine learning methods where the training of the model is essential. Some methods are specific to special varieties of short text such as email and customer queries whereas some are specific to specific domains such as medicine or law. However pre-trained methods of text summarization required very huge datasets for training, highly powerful systems for running and are only good as the data that they are trained on. In other words, they are constrained to domain and language it was trained for and cannot be used for arbitrary text. For example, a machine-learning-based summarization module trained for medical research text cannot be used for summarization of texts from physics though they are scientific texts and belong to the same domain. There is a need for a method in which the summarization method can be applied to a large variety of text and can be extended to other languages. The method needs to be definitive, repeatable and transparent so that it can be easily adapted to other domains and languages. However, minimal research has been carried out in this field.

## 1.2 Contributions

The major contribution of this work is the presentation of a practicable text summarization algorithm that can be applied to scientific text from various domains. This work has been extended from the work by Benbrahim and Ahmad [2]. The salient features of the *Intelligent Text Summarization algorithm* are as follows.

- The algorithm is designed in such a way so that it can accept arbitrary English scientific text from various domains and autonomously produce summaries. The

core algorithm identifies the key terms which in turn are used for summarization in a language-agnostic manner.

- The algorithm makes use of the finding that, key ideas in the form of terms and phrases are repeated in scientific texts. Hence the key terms can be identified using simple mechanical computation.
- The algorithm is definitive and repeatable. The results produced are consistent and statistically verifiable.

In addition to the core algorithm, this work also makes the following contributions.

- i. An algorithm for statistically verifying the goodness of the summary that is produced. This is an important contribution since the evaluation of summary is a human-intensive process and is difficult to be carried out on different domains on a large scale.
- ii. The algorithms for summarization and statistical verification of goodness of summary is implemented as a working software with a client user interface that can be run on both Microsoft Windows and Apple OSX operating systems. The server is implemented using Representational State Transfer (ReST) interface that serves requests securely over the Hyper-Text Transfer Protocol (HTTP) that can be used by other software systems as well. The software is named '*Curukka*' which means abstract in a Dravidian language called Tamil - one of the oldest languages in the world. Its logo is given in figure 1.3



Figure 1.3: The logo for Curukka Text Processing Pipeline



- iii. The software implementation also consists of data exporters in Comma Separated Value (CSV) format for the metrics and other statistical computations produced which can be imported into other systems.

## 1.3 Structure of the Dissertation

The remainder of the dissertation is organized as follows. The motivation for the work and the literature review covering the latest related works, existing approaches, and evaluation techniques is explained in chapter 2. The methods used for summarization and evaluation of the generated summaries are explained in chapter 3. Following the explanation of the methods in chapter 3, the systematic collection of a dataset for evaluation, the implementation of the algorithms as a working software along with its the design specifications, case studies for evaluation of the goodness of the summaries produced and the performance of the system is explained in chapter 4. The limitations of the work, conclusion and the suggestions for future work are given in chapter 5. The detailed notes on the setup of the intelligent text summarization software named 'Curukka' and step-by-step working instructions are given in the appendix section of the dissertation.

The research design plan is graphically showcased in figure 1.4

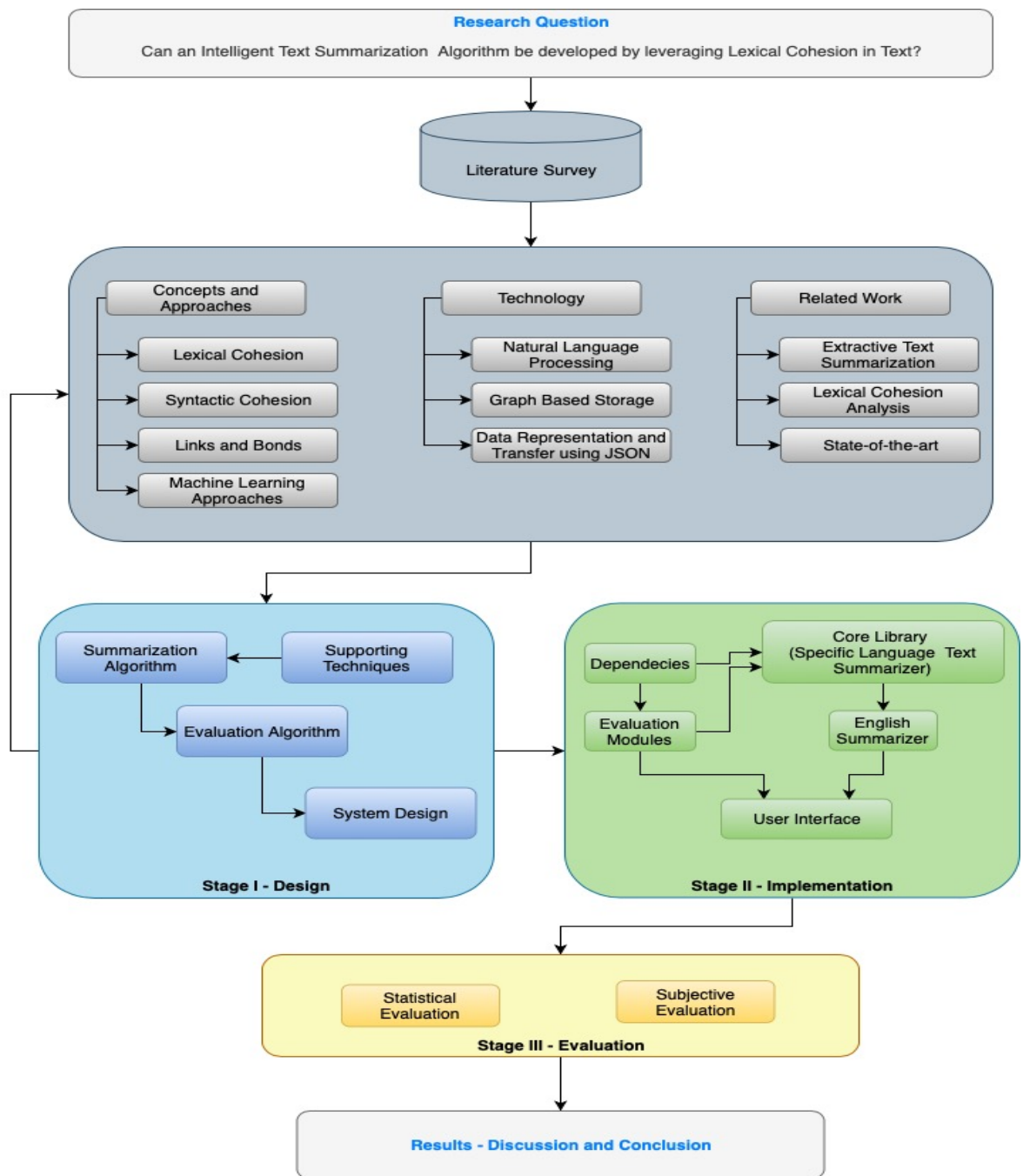


Figure 1.4: Graphical representation of the research design plan

# Chapter 2

## Motivation and Literature Review

Chapter 1 provides the introduction to this work. This chapter explains the motivation (see section 2.1) behind this work and the literature review (see section 2.2) of the approaches, techniques and related work along with the dataset and evaluation techniques used in them.

### 2.1 Motivation

Automatic text summarization can be useful for a myriad of purposes. Moreover, manual summarization is expensive and dependent upon the the writing skills of the author of the article or the technical knowledge of a professional text summarizer.

Scientific texts are produced in abundance in various domains and effective summarization of the texts is a very important task. Though some of the scientific texts in the recent past come with abstracts, the quality of scientific writing is deteriorating over last two decades or so, especially due to 'bad writing' (See for example, Freeling et al 2019 [17]); and some authors have stressed that abstracts can be improved much further to help the reader in selecting a given article and then reading it - the abstract being a very important in this context (See for instance Plavn-Sigray et al 2017 [18]). Some of the desirable features of automatic text summarization are: it should produce a summary without the loss of key information from the text; should be able to summarize an arbitrary scientific text; should be able to summarize text from various domains; and should be transparent, definitive and repeatable.

When non-narrative scientific text is considered, they tend to follow a specific style that is used to persuade the reader of the importance of concepts and arguments presented in the paper - the term *theme* is used in this context which is used to suggest how a writer/speaker identifies the relative importance of the subject matter of the paper [19]. The authors tend to introduce a set of domain-specific terms. Throughout the text, the authors repeat the domain-specific terms or open-class words to explain and elaborate on the topic. The topic can be repeated by simple repetition of the term itself, its morphological variations or as complex repetitions as is called lexical cohesion. The topics may also be linked together by conjunctions, including *and or but*, pronominal references, for instance *it* instead of the *name of an object*, ellipsis, for example *cloud* instead of *cloud computer* or *cloud computing*. The linkage provided by conjunctions, pronouns, and strategies for elliptical paraphrase, are part of the grammar or syntax of the language in which a text is written. These grammatical or more precisely syntactic links, provide syntactic cohesion and usually do not carry as much information about the content as do lexical cohesion. This work focuses mainly on lexical cohesion. Using lexical and syntactic cohesion, the sentences in the text are linked together to form the complete text. The various forms of lexical and syntactic cohesion are shown in figure 2.1 and are studied in detail by Halliday and Hasan [20]. The difficulty of using syntactic cohesion and the usage of lexical cohesion where certain lexical features of the text connect the sentences in the text is studied theoretically in detail from the perspective of linguistics by Michael Hoey [21].

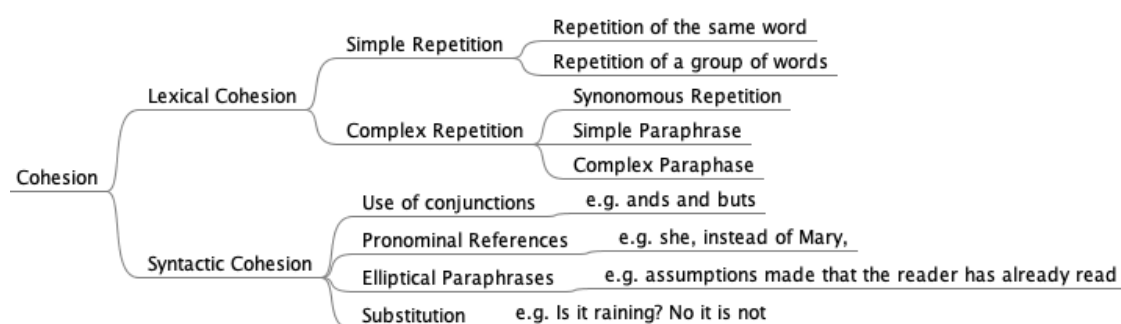


Figure 2.1: Different types of lexical and syntactic cohesion

Computers are good at performing primitive tasks such as addition, subtraction,

multiplication, and division along with executing primitive functions such as searching, sorting and merging. If an algorithm can leverage the concept of lexical cohesion to identify the important sentences of a text in such a way that it can perform summarization by making use of simple mechanical computation using the primitive functions, it would lead to an algorithm that is elegant, practicable, versatile, definitive, and will satisfy the desirable features of automated text summarization.

The work by Paice and Jones (1993) [22] attempts to identify important topics in highly structured technical documents effectively combining the tasks of indexing and abstracting. The work by Benhrahim and Ahmad (1995) [2] leverages the concept of lexical cohesion to produce summaries by forming a directed graph of sentences. The work making use of simple repetition of key terms in the text to create a graph of sentences from which the summaries are derived. However, very little work has been carried out on text summarization using lexical cohesion. The work by Erkan and Radev (2004) [23] is a recent related work that uses a graph-based lexical centrality by computing the intra-sentence cosine similarity.

The need for an effective text summarization algorithm that uses the concept of lexical cohesion that can be implemented by leveraging primitive computer functions along with the relative lack of research in this field is the motivation for this work.

## 2.2 Literature Review

This section includes the studies performed by various contemporary researches which is the key motivation and helps to provide a comprehensive view of the current state of the research.

The automated text summarization topics and approaches the have been studied in this work are shown in figure 2.2. The green links show the approaches that have been used in this work.

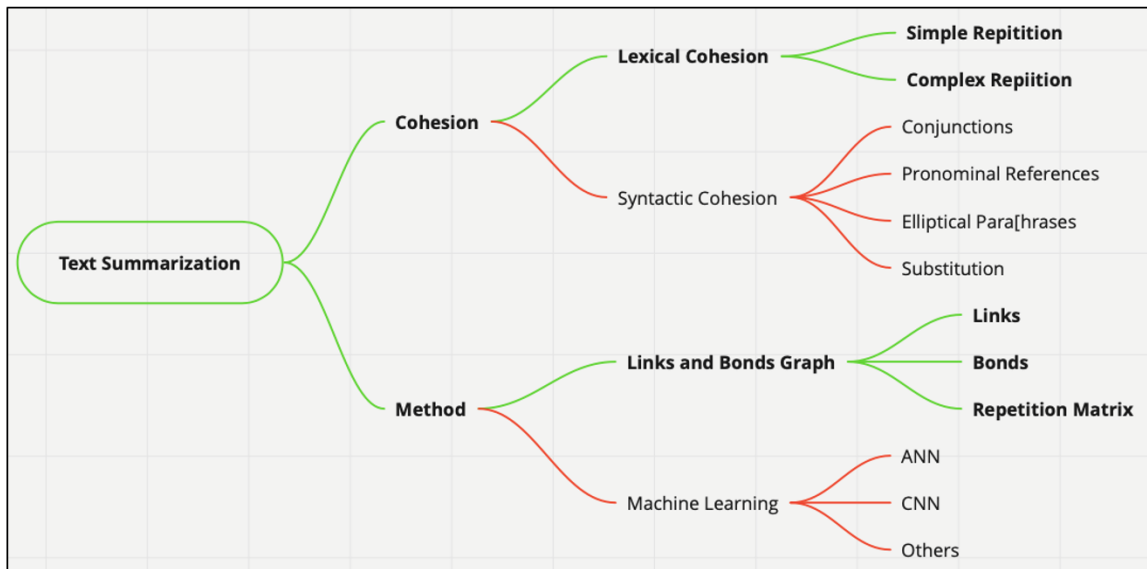


Figure 2.2: Summary of studied approaches (Green links show the used approaches)

Figure 2.3 shows the categorized literature review carried out in this work. The literature review is primarily categorized into works that make use of cohesion in text and the various methods used for summarization. In terms of the work leveraging cohesion in text, it is further classified into works that leverage lexical cohesion and the works that leverage syntactic cohesion. In terms of the methods, it is further classified into works that make use of sentence graphs for text summarization and those that make use of machine learning methods for text summarization.

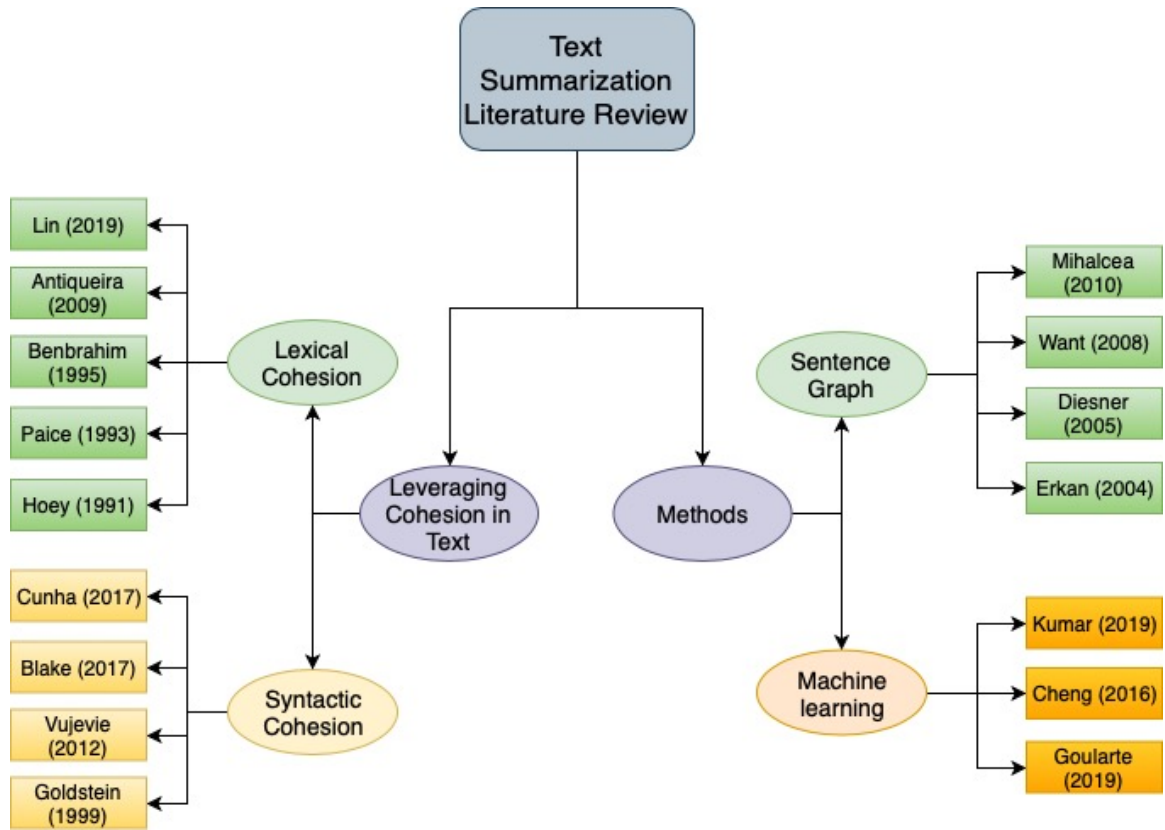


Figure 2.3: Overview of the literature survey on text summarization techniques and references for various modes of operation

### 2.2.1 Lexical Cohesion

On the basis of phonology, lexicography and grammar levels, the elements share semantic relationship as proposed in [24]. This cohesive approach based upon the extension of gratification and comprehension has been an influential premise in simultaneous interpretation of the text. Thus when the text is semantically connected with repetition and other types of cohesions, only then it makes semantic sense of communication in the audience. Another approach has explored selective retention of complex repetitions in [25] using metrics of complex networks. A graph is defined with edges corresponding to the meaningful nouns and nodes for the sentences. Afterwards, k-cores, d-rings, length of shortest paths, degree of nodes has been explored using network concepts. Brazilian Portuguese texts has been analyzed for the performance analysis to support

the belief of automatic extractive summary.

### **2.2.2 Syntactic Cohesion**

Discourse segmenter has been developed for parsing for discourse segmentation in [26]. Syntactic and lexical information has been used to transform the rules of one language to another (Spanish to Catalan) using Rhetorical Structure Theory. Result analysis has been performed using standard corpus and manual texts. In [27], a news article based analysis of the text summaries based on sentence ranking is proposed. The ranks are assigned by considering linguistic features and statistical metrics. Precision-recall curves with a random sentence selection baseline model has been studied for evaluation to emphasize the utility of corpus-dependent standards of text summary, careful crafts of long queries and compression ratio deliverables.

In [28], semantic capturing of elliptical noun phrases have been studied. The elliptical coordinated compound nouns are often used for space efficiency of a research paper to avoid the mis-intrepretation of a disease or a body parts for phrases such as “lung or breast cancer” for example. Extensive experiments have been performed using full texts to quantify the impact of coordinated ellipses by considering backward, forward, complex ellipses, generate-and-test phenomenon for candidate expansions and leveraging syntax. Another method for substitution and ellipsis as cohesive linguistic devices has been studied in [29]. This study has explored the ellipsis and substitutions as linking mechanisms of textual cohesive relationships among sentences. Repetition of sentences should be avoided to generate a better cohere version of the text using illustrated examples on linguistic devices of substitution and ellipsis.

### **2.2.3 Links and Bonds**

Link and graph based ranking algorithm for text processing has been invented in [30]. Based upon natural language, it considers text plurality using the graph nodes. This follows by searching connecting links among plurality of text nodes. It includes other features such as sentence/keyword extraction, semantic disambiguation and graphical visualization of text collection. In [31], a novel text analysis has been studied to reveal social structures using semantic networks or maps. Bonds among events, organizations and people have been observed using elements of the text and meta-matrix approach



based upon an ontology. An interesting example for illustration using West Bank and terrorist groups in operation and how to discover them, has been studied using social structures of covert networks. Symmetric matrix factorization and sentence level semantic analysis text summarization has been proposed in [32]. It begins with similarity calculations among sentences to generate the matrix, followed by symmetric matrix factorization to group sentences in clusters. Symmetric matrix clustering works has been found similar to normalized spectral cluster. DUC2005 and DUC2006 data sets have been considered for numerical case study.

#### 2.2.4 Soft-computing and fuzzy systems

In [33], text summary has been analyzed for text assessment in expert systems. This system is developed using fuzzy rules to specify extracted feature variety to compute the information which is most important. Dimensions has been reduced using correlation among features on the dataset of Brazilian Portuguese texts. Method was compared with Model and Sentence, Score, naive baseline with the help of ROGUE measure and yielded better F1-score. An artificial neural network based text summarization has been proposed in [34]. It is based on the features of continuous sentences and data-driven without a need to rely on human engineering. It extracts words and sentences to devine various classes of summary models. The illustration of this unsupervised learning has been performed on two datasets for linguistic annotations. A deep learning model based on convolutional neural network for sentiment analysis for text analysis has been studied by [35]. It used ontology to generate semantic features, Word2Vec for corpus conversion and CNN for opinion mining. Performance comparison has been quantified using F-measure, recall, precision and accuray so support the belief in the proof of concept of prototype model.

### 2.3 Conclusion

The various literature of automated text summarization studied is compared illustrating the method, idea and the dataset used for evaluation in table 2.1

This work considers the research works studied with careful consideration for the design of the *Intelligent Text Summarization* algorithm. The literature is used to

Table 2.1: Comparative analysis of automated text summarization techniques with highlights and dataset

SR	REF.	METHOD	IDEA	DATA
1	Lin (2019) [24]	Simple Repetition	Semantic relationships, cohesion using gratification and compression	Random text
2	Antiqueira (2009) [25]	Complex Repetition	Graph matrix with sentence nodes and meaning as nodes	Brazilian Portuguese texts
3	Cunha (2017) [26]	Conjunctions	Rhetorical structure theory for tranlation of Spanish to Catalan	News articles
4	Goldstein (1999) [27]	Pronomial References	Sentence ranking using linguistic and statistical features	20 sets of 50 docs from TIPSTER 1988-1992
5	Blake (2017) [28]	Ellipitical Paraphrases	Ellipitical coordinated compound nouns, backward, forward, complex ellipses.	Journal of Biological Chemistry
6	Vujevic (2012) [29]	Substitution	Ellipsis and substitutions to link textual cohesions	Online news articles
7	Mihalcea (2010) [30]	Links	Connecting pluraity of text nodes using sentence and keyword extraction, graphical visualiation and semantic disambiguation	Search engines (Google)
8	Diesner (2005) [31]	Bonds	Bonds among social structures using semantic networks, meta-matrix approach using ontology	West bank and terrorist groups discovery
9	Wang (2008) [32]	Repetition Matrix	Similarity calculations among sentences using symmetric matrix factorization	DUC2005 and 2006
10	Cheng (2016) [34]	ANN	Features of continuous words are fed into unsupervised neural network	191 texts collected at CASOS
11	Kumar (2019) [35]	CNN	Sentiment analysis using Word2Vec, corpus creation, CNN for opinion mining (good or bad)	moviereview and hotelreview
12	Goularte (2019) [33]	Fuzzy	Fuzzy rules for feature variety to compute information, dimension reduction using correlation analysis	Brazilian Portuguese

understand the various methods explored by researchers for automated text summarization and evaluation. The literature review also enables this work to be done in an informed manner considering the pros and cons of the existing works or overcoming the shortcomings. This work extends on the work by Benbrahim and Ahmad [2]. A major contribution is the design of an intelligent text summarization algorithm which understands the natural language present in the scientific text, intelligently and autonomously identifies key terms in the text and uses the key terms to create a summary of the text contributing to artificial intelligence. The method is explained in chapter 3.

# Chapter 3

## Method

### 3.1 Introduction

Text analytics has been a growing branch of natural language processing, which is itself a sub-branch of Artificial Intelligence. The works in the field of natural language processing have opened up many unexplored fields. Areas like speech recognition rely heavily on speech to text and then natural language processing of the text to analyze its meaning. The role of text analytics in text indexing is of paramount importance. Topic modeling and key phrase extraction play a vital role in search engines.

The treatment of texts within text analytics relies generally on the grammatical structure of the language of the text. Part of speech taggers is used in various text analytics methods to understand the grammatical structure of the text and make effective use of it. Natural language processing engines like the ones from Stanford [36] have gained popularity and is used widely. However, seldom attention is paid to how authors introduce keywords, how they elaborate keywords, and how they use keywords in complex sentences. The use of keywords for representing core ideas has proved to be instrumental in domain-specific texts [37]. Algorithms for text linguistics have often paved the way in this respect. Texts written by scientists and engineers pose significant problems in terms of indexing on the one hand and summarization on the other. Moreover, specialist texts are jargon-rich and cannot be dealt with easily by novices in the domain and nor indexing experts. People with a deep understanding of the domain, expertise in comprehension and skill in precis writing is required to create

meaningful summaries of scientific texts. Text summarization strategies also rely on repetition of tokens and scientific messaging relies on repetition, which is our key finding thus far. The algorithm presented in this chapter is based on this understanding. The motivation for this work and the literature review is given in chapter 2.

The language used in scientific texts is very peculiar, it has been established in genre analysis of language under the domain of text linguistics [38]. The language used in scientific texts is specific to the domain and uses a large set of domain-specific terms. Some terms are introduced by the authors and then it becomes a part of the language. These domain-specific terms have a different meaning in the domain-specific text and the frequency of usage varies sharply from the general language. For example, the word 'nucleus' has a different meaning in general English which means 'center', a different meaning in Biology meaning, 'the center part of a cell' and a very varied meaning in Physics meaning, 'the core of an atom which consists of protons and neutrons'. The usage of 'nucleus' to mean center is comparatively less in common English, however, when it comes to Biology or Physics, the word 'nucleus' is more generally used and hence the relative frequency (the frequency with respect to the text) increases. This is one of the core ideas on which the text summarization algorithm is devised in this chapter.

Figure 3.1 shows the core steps in the 'Intelligent Text Summarization' algorithm proposed in this work and explained in the later section of this chapter. There are a total of eight distinct steps in the whole process labeled from 'a' to 'h'.

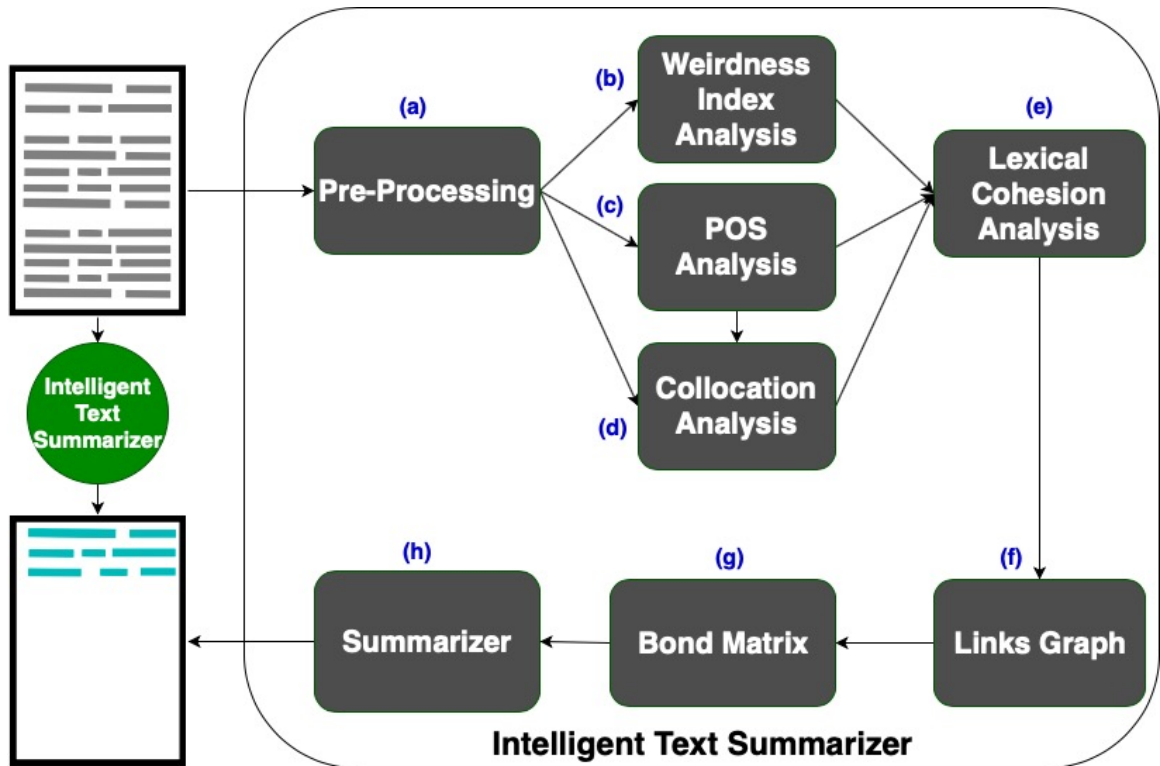


Figure 3.1: The proposed flow of the algorithm with the core methods

## Key Terms Used

The key terms involved in the research have been defined below.

1. **Lexical Cohesion** : The action or condition of cohering. cleaving or sticking together
2. **Link** : Two sentences are said to be linked when they have a common key token in them.
3. **Bond** : Two sentences are said to be bonded when more than a certain (default 3) number of links exist between them.
4. **Repetition Matrix** : A nxn matrix of bonds formed between sentences.
5. **Open Class Words** : Domain Specific Words. E.g. Computer, Software, Modem.

6. **Closed Class Words** : Words Common to the Language for example the, and, which, etc.

The pre-processing of the text is explained in section 3.2. The intelligent text summarization algorithm is explained in section 3.3 which includes key term extraction (see section 3.3.1), sentence linking (see section 3.3.2), bonding (see section 3.3.3), the calculation of in-out bond ratio (see section 3.3.4) and the method for summary generation (see section 3.3.5). The evaluation methods used for automated evaluation of the summaries produced is explained in section 3.4, consisting of three methods for automated evaluation namely readability index (see section 3.4.1), correlation of relative frequency (see section 3.4.2) and cumulative relative frequency comparison of top open-class words (see section 3.4.3). The conclusion for the chapter is provided in section 3.5.

## A Sample Text

A computer system has three main components: hardware, software, and people. The computer can be a tool for just about anyone from a business person, to an artist, to a housekeeper, to a student - an incredibly powerful and flexible tool.

The equipments or physical components associated with a computer system is called computer hardware system. These include the mouse, keyboard and the central processing system. Computer Software is a set of instructions fed in by people that tells the computer hardware what to do. People, however, are the most important component of a computer system - people use the power of the computer for some purpose.

Software is actually a computer program. To be more specific, a program is a set of step-by-step instructions that directs the computer to do the tasks you want it to do and to produce the results you want. A computer programmer is a person who writes programs. Most of us do not write programs, we use programs written by someone else. This means we are users - people who purchase and use computer software.

Figure 3.2: Sample scientific text as an example

A short sample scientific text about a computer [39] is shown in figure 3.2. This text will be used to illustrate the working of the algorithm in the various steps detailed in this chapter.

## 3.2 Pre-Processing the Text

The pre-processing stage is not a direct part of the core algorithm. However, it is an essential step since the text that has to be given as input for automatic text summarization should be in a machine-readable format that is compatible with the other components for the successful execution of the algorithm.

In the pre-processing stage, two important pre-processing steps are done. The first is to check for the encoding of the text. It is important that the encoding of the text is uniform so that the characters in the text are not misinterpreted. This is a vital check since the digitized documents when used in different computer operating systems can be encoding to the native encoding format of the operating system. There are also cases where the encoding within a text may be mixed. Hence, the encoding of the input text has to be made uniform. Based on the experiments carried out, the character encoding set ISO-8859-1 [40] is chosen as the base format. All the input text is converted to the ISO-8859-1 encoding format being fed into the summarization algorithm.

Another pre-processing stage that is carried out is the removal of ASCII control characters and HTML tags that are present in the text. The ASCII control characters render non-humanly-visible changes to the text that change the natural working of the algorithm when implemented as computer software. The HTML tags in the text, if present, are used for markup when the text is displayed on the web and renders no meaningful information to the text in the perspective of summarization as per the algorithm showcased in this work. Hence, the HTML tags are removed.

Once the pre-processing of the text is complete, the text is passed on to the core algorithm for summarization.

## 3.3 Intelligent Text Summarization Algorithm

The *Intelligent Text Summarization* is a nine-step algorithm. The algorithm starts by identifying the key terms of the text. Various methods such as weirdness index, Part-Of-Speech (POS) analysis and collocation analysis is used to identify the key terms.

The next step involves the lexical cohesion analysis of the text by calculating their relative frequency and creating a signature vector (vector of words uniquely represent-



ing the text that is created based on the frequency distribution) based on the frequency. Once the signature vector is created based on the frequency distribution, the sentences are linked together. Based on the links between sentences, a directed graph of sentences is created. When the number of links between two sentences is greater than a given threshold called *bond strength*, a bond is established between the sentences.

The links between sentences and the bonds are represented as matrices called the link matrix and bond matrix respectively. Once the bond matrix is created, the ratio of inbound links and outbound links for each sentence is calculated based on the number of incoming bonds and outgoing bonds respectively. The density of repetition linkage between the sentences is a measure of the closeness of the sentences and a statistical method of determining that the sentences are a part of the common theme that is presented in the text.

The ratio of the inbound bonds and outbound bonds is used to determine whether a sentence is an *OPENING* sentence - a sentence that opens the topic in the text, *MIDDLE* sentence - a sentence that is used for elaboration and contains the meaningful middle part of the text, *CLOSING* sentence - a sentence that concluded a topic in the text and *MARGINAL* - a sentence that is used for explanation of the topics in the text but do not necessarily is core to the text. The idea is that a marginal sentence can be safely eliminated without losing any information from the text.

The final step of the algorithm is to present the summary by organizing the identified sentence in a manner that is easy to be read and understood by the reader. The 9 steps of the algorithm is shown in figure 3.3

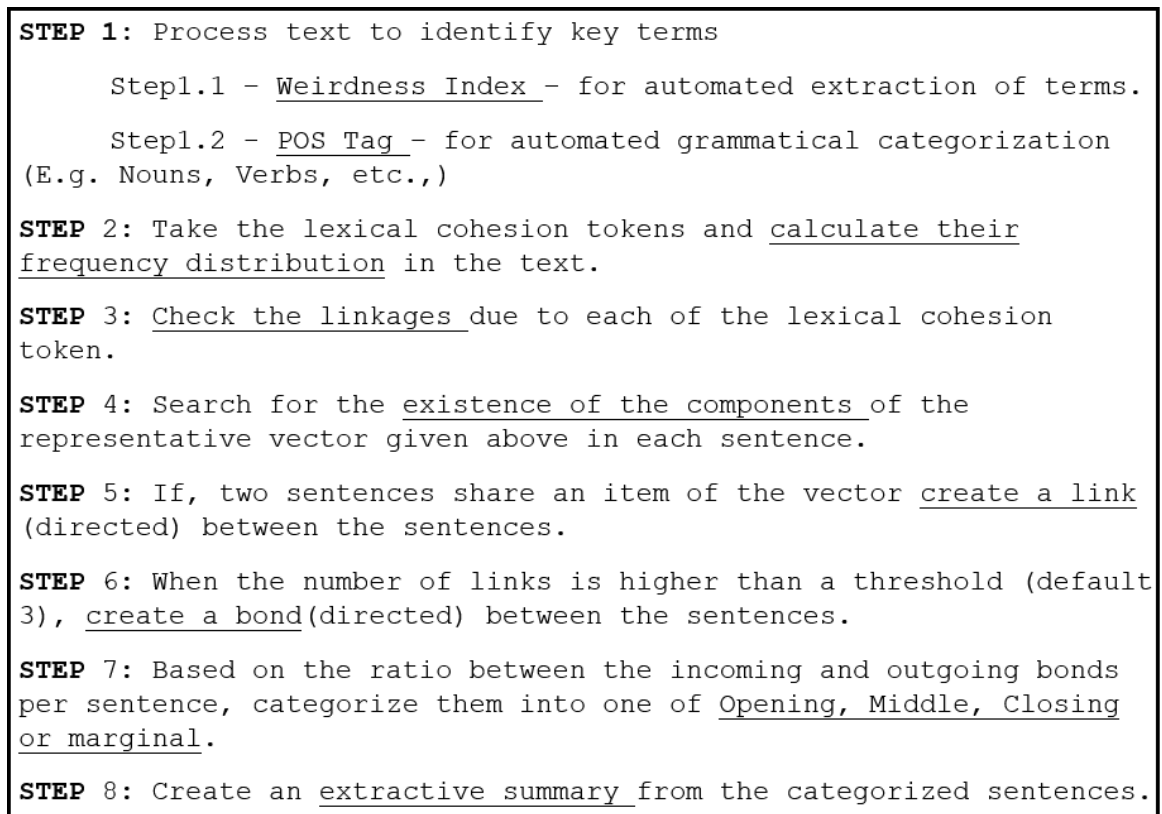


Figure 3.3: Algorithm of the proposed text summarization technique - Intelligent Text Summarization

The following five sub-sections provides detailed method explanation for each of the important section of the algorithm.

1. Key Terms Identification (see section 3.3.1)
  - (a) Weirdness Index Calculation
  - (b) Collocation Analysis
  - (c) Part-of-speech Analysis
2. Sentence Linking and Link Matrix (see section 3.3.2)
3. Sentence Bonding and Bond Matrix (see section 3.3.3)
4. Sentence Categorization using in-out bond ratio (see section 3.3.4)

## 5. Summary Generation (see section 3.3.4)

### 3.3.1 Key Terms Identification

Automatic key term identification is the most vital part of the intelligent text summarization algorithm showcased in this work. The key terms identified is used for the creation of the signature vector that is in turn used for the linking and bonding of sentences from which the summary for the text is generated.

Automatic Term Identification has been an important part of text retrieval systems and various methods are employed for automatic key terms extraction [41]. In this work, the primary method used for the key term extraction is *Weirdness Index method* which enables autonomous and intelligent key terms extraction. Besides, the part-of-speech (POS) tagging method is used to enhance the algorithm. However, the part-of-speech tagging method is dependent on the language and needs POS tagger trained in the language to tag the text with its part-of-speech. Finally, another improvement brought about in this work compared to the previous works [2] [22] [23] is the incorporation of collocation analysis to find collocates - co-located words that enabled the algorithm to perform compound words repetition analysis providing better automated text summarization.

#### Weirdness Index Calculation

Weirdness Index Calculation method for the identification of key terms in a relative frequency calculation and corpora comparison method to identify domain-specific terms or open class works in a simple to use and easily computable manner [37] [42].

The method does not need the use of any pre-training and depends on a simple mathematical calculation of relative frequencies of the terms in the text with the terms in a reference corpus. A reference corpus is a collection of a diverse range of texts from various genres, carefully selected and organized to represent the use of words in the given language. Usually, the reference corpus is maintained by a consortium and is updated regularly to reflect the current usage of words in the language.

In this work, since English language scientific text is used, two English language reference corpora are used. First is the Open American National Corpus (OANC) is a corpus of English language text containing text that uses American English main-

tained by the Linguistic Data Consortium. The corpus consists of 22 million words of written and spoken American English across various genres including literature, dailies, weeklies, and emerging texts like tweets and web data [43].

Similar to the Open American National Corpus, the British National Corpus (BNC) is a carefully selected corpus of English texts are written or spoken in British English. The British National Corpus contains 100 million words from varied genres of text and stands as a representative sample of the spoken and written British English [44].

OANC and BNC are used in this work to compare the frequencies of the selected words in the representative sample of the language. Both OANC and BNC are available as machine-readable formats as Tab Separated Value (TSV) files. The TSV file consists of the four fields namely: sort-order, frequency ( $freq(W_R^k)$ ), word ( $W_R^k$ ), and word-class. The sort-order is the rank of the word based on the frequency, the frequency is the number of occurrences of the word in the corpus, the word represented the word itself and the word-class gives the part-of-speech tag of the word [45]. A sample snapshot of a set of TSV in the corpus is given in figure 3.4

```
5 2186369 a det
2107 4249 abandon v
5204 1110 abbey n
966 10468 ability n
321 30454 able a
```

Figure 3.4: A sample snapshot of the BNC corpus with words represented as TSV

From the data available in the corpus, the relative frequency of each word in the corpus is calculated using the algorithm 1.

**Result:** Corpus updated with the relative frequencies of each word

```

1 Load Corpus into memory;
2 total  $\leftarrow$  0;
3 foreach  $W_R^k$  in Corpus do
4   |  $total \leftarrow total + freq(W_R^k)$ 
5 end
6 foreach  $W_R^k$  in Corpus do
7   |  $RelativeFreq(W_R^k) \leftarrow freq(W_R^k)/total;$ 
8 end

```

**Algorithm 1:** Algorithm to calculate relative frequency of terms in the reference corpus

Both OANC and BNC are referred commonly as the reference corpus in this work and is represented using the notation  $CORPUS_R$ . The corpus of tokens that is created from the input scientific text is referred by the notation  $CORPUS_S$ . Similar to the reference corpus, the specific text corpus is also processed to get the frequency ( $freq(W_S^k)$ ) of each word ( $W_S^k$ ) in the corpus.

The weirdness index of a word is calculated as the ratio of the relative frequency of the token  $W^k$  in the scientific text corpus  $CORPUS_S$  to the relative frequency of the token in the reference corpus  $CORPUS_R$ . The formula for weirdness index calculation is given in equation 3.1.

$$Weirdness = \frac{\frac{freq(W_S^k)}{N_S}}{\frac{freq(W_R^k)}{N_R}} \quad (3.1)$$

where

$freq(W_S^k)$  = Frequency of the word ( $W^k$ ) in the specific text corpus  $CORPUS_S$   
 $N_S$  = Total number of words in the specific text corpus  $CORPUS_S$   
 $freq(W_R^k)$  = Frequency of the word  $W^k$  in the reference corpus  $CORPUS_R$   
 $N_R$  = Total number of words in the reference corpus  $CORPUS_R$

## Dealing with the shortcomings of Weirdness Index Method

Though weirdness index is a very good method of identifying the open-class words from the text, it suffers from a set of disadvantages. One, if the word is not updated in the

reference corpus, then weirdness of the word will increase to infinity providing a false positive. Similarly, if the word in the text has a spelling mistake, the word will not be available in the reference corpus, hence the weirdness index will increase to infinity again providing a false positive.

To overcome this situation, the weirdness index method is augmented by making combined use of  $z$  – score of the frequency of the word ( $freq(W_S^k)$ ) in the specific text corpus  $CORPUS_S$  and the  $z$  – score of the weirdness index of word ( $weirdness(W_S^k)$ ) in the specific text corpus  $CORPUS_S$ . The  $z$  – score is a simple measure of the number of standard deviations the given items is over or below the mean. The  $z$  – score is defined by the equation 3.2.

$$z - score = \frac{x - \mu}{\sigma} \quad (3.2)$$

where

$x$  = Frequency of the word

$\mu$  = Mean of the frequency of the word

$\sigma$  = Standard Deviation of the frequency of the word

Referring 3.2, the *frequencyz – score* can be calculated as follows

$$frequencyz - score = \frac{freq((W_S^k)) - mean(freq(W_S^k))}{sd(freq(W_S^k))} \quad (3.3)$$

where

$freq((W_S^k))$  = Frequency of the word ( $W_S^k$ ) in the specific text corpus  $CORPUS_S$

$mean(freq(W_S^k))$  = Mean of the frequency of the word in the specific text corpus  $CORPUS_S$

$sd(freq(W_S^k))$  = Standard Deviation of the frequency of the word in the specific text corpus  $CORPUS_S$

Similarly, the *weirdnessz – score* can be calculated as follows

$$weirdnessz - score = \frac{weirdness((W_S^k)) - mean(weirdness(W_S^k))}{sd(weirdness(W_S^k))} \quad (3.4)$$

where

$\text{weirdness}((W_S^k))$  = Weirdness of the word ( $W_S^k$ ) in the specific text corpus  $CORPUS_S$   
 $\text{mean}(\text{weirdness}(W_S^k))$  = Mean of the weirdness of the word in the specific text corpus  $CORPUS_S$   
 $\text{sd}(\text{weirdness}(W_S^k))$  = Standard Deviation of the weirdness of the word in the specific text corpus  $CORPUS_S$

Based on equation 3.2, the  $z$  – score for frequency (see equation 3.3) and weirdness (see equation 3.4) is calculated for each token in the specific language text. Only those tokens for which the  $z$  – score for both the frequency and weirdness is greater than one is chosen as key terms. This avoids false positives with tokens that are not available in the reference corpus and those tokens which have spelling mistake. The selection criteria is depicted graphically in figure 3.5

Z-Score		Weirdness	
		$\geq 1$	$< 1$
Frequency	$\geq 1$	Candidates	Non-Candidates
	$< 1$	Non-Candidates	Non-Candidates

Figure 3.5: Graphical representation selection criteria of terms using the Z-Score of the frequency and weirdness index

The algorithm for the selection of key terms using the Weirdness Index method is given in algorithm 2.

**Result:** Key Terms selected from the text using Weirdness Index Method

```
1 specificLanguageText ← Load Specific Text Corpus into memory;  
2 tokenizedText ← tokenize(specificLanguageText);  
3 candidateTerms ← Empty Set;  
4 weirdnessZScoreMap ← Empty Map;  
5 frequencyZScoreMap ← Empty Map;  
6 foreach token in tokenizedText do  
7   | weirdnessZScoreMap[token] ← weirdnessZScore(token)  
   | frequencyZScoreMap[token] ← frequencyZScore(token)  
8 end  
9 foreach token in tokenizedAnnotatedText do  
10  | if weirdnessZScoreMap[token] > 1 AND frequencyZScoreMap[token] >  
   | 1 then  
11  |   | candidateTerms ← append(token);  
12  | else  
13  |   | Discard Token;  
14  | end  
15 end
```

**Algorithm 2:** Algorithm to select key terms using Weirdness Index Method

As per algorithm 2, then input text for which the key terms have to be extracted is first tokenized. Tokenization is a process of converting the text into individual words. For each token in the input text the frequency z-score (see equation 3.3) and the weirdness z-score (see equation 3.4) is calculated. Once the frequency and weirdness z-scores are calculated for all the tokens, tokens are once again iterated and only the tokens for which the z-scores for both the frequency and weirdness is greater than zero is considered as candidate terms (key terms).

The design specification for the selection of the terms using the weirdness index method is given as a Unified Modelling Language (UML) flow-chart in the figure 3.6



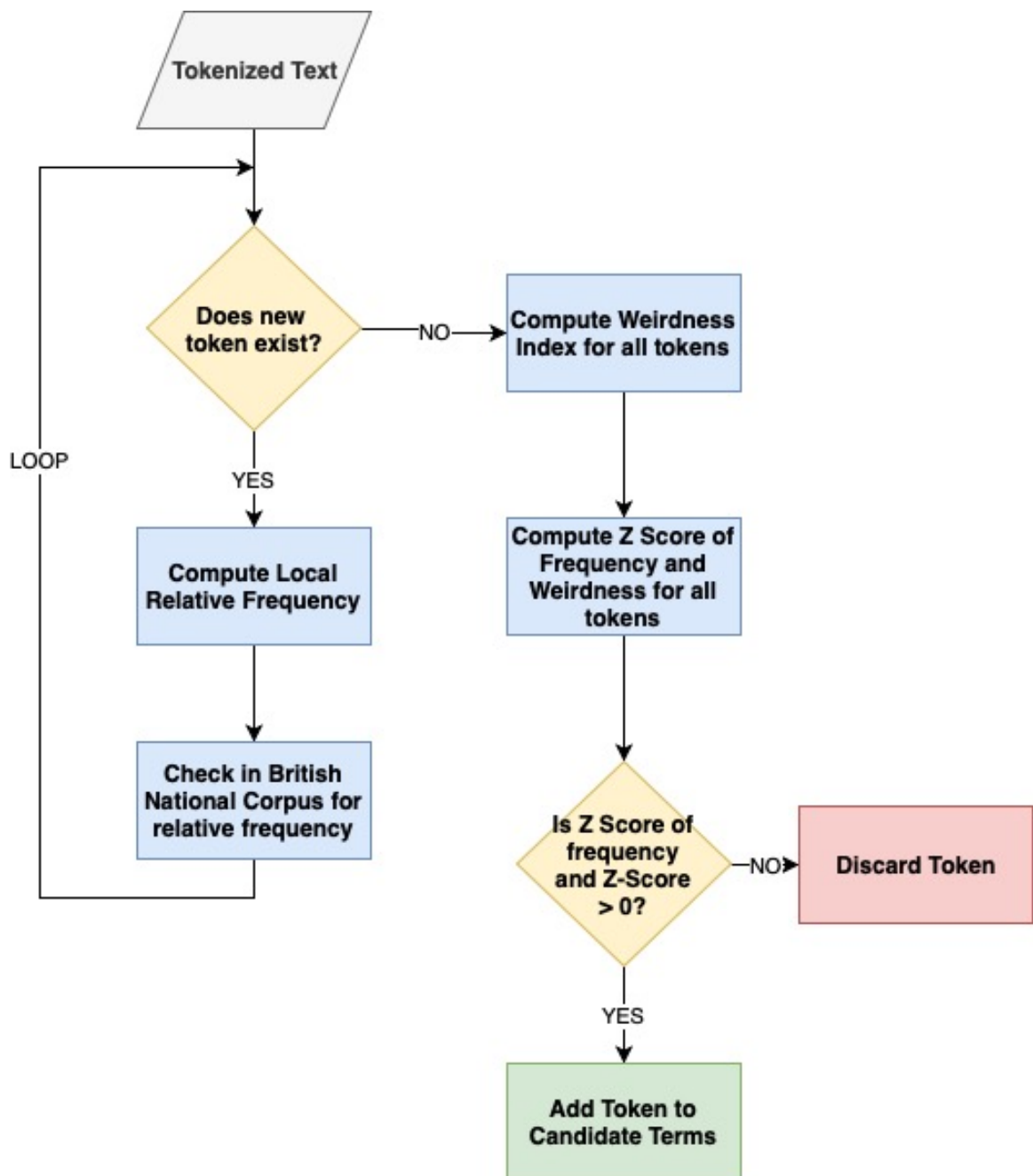


Figure 3.6: Design Specification of the selection of terms using Weiridness Index represented using UML Flow-chart

## Part-of-speech Analysis

The part-of-speech (POS) analysis is an add-on to key term extraction. The key term extraction is capable of working without the use of a POS tagger and is not a must-be dependency for the algorithm. However, it is used to enhance the set of key terms by annotating the input scientific text with the part-of-speech and adding the nouns from the specific text corpus *CORPUS<sub>S</sub>* into the candidate terms. The POS taggers are available as of-the-shelf modules that can be used to annotate the text with POS tags.

In this work, the StanfordNLP POS tagger is used. The StanfordNLP POS tagger is widely used in the industry and academia alike [36].

The POS tagger tags the text with various POS tags. The various POS that will be used to annotate the text by the POS tagger is given in table 3.1.

Table 3.1: Part-of-speech tags used by POS tagger to annotate text

Sr	POS Tag	Explanation
1	CC	coordinating conjunction
2	CD	cardinal digit. Eg. '1'
3	DT	determiner. Eg. 'The'
4	EX	existential there (like: "there is" ... think of it like "there exists")
5	FW	foreign word
6	IN	preposition/subordinating conjunction
7	JJ	adjective Eg. 'big'
8	JJR	adjective, comparative Eg. 'bigger'
9	JJS	adjective, superlative Eg. 'biggest'
10	LS	list marker Eg. '1')
11	MD	modal Eg. 'could', 'will'
12	NN	noun, singular Eg. 'desk'
13	NNS	noun plural Eg. 'desks'
14	NNP	proper noun, singular Eg. 'Harrison'
15	NNPS	proper noun, plural Eg. 'Americans'
16	PDT	predeterminer Eg. 'all the kids'

17	POS	possessive ending Eg. parent's
18	PRP	personal pronoun Eg. 'I', 'he', 'she'
19	PRP\$	possessive pronoun Eg. 'my', 'his', 'hers'
20	RB	adverb very, Eg. 'silently'
21	RBR	adverb, comparative Eg. 'better'
22	RBS	adverb, superlative Eg. 'best'
23	RP	particle Eg. 'give up'
24	TO	to go 'to' the store.
25	UH	interjection errrrrrrm
26	VB	verb, base form Eg. 'take'
27	VBD	verb, past tense Eg. 'took'
28	VBG	verb, gerund/present participle Eg. 'taking'
29	VBN	verb, past participle Eg. 'taken'
30	VBP	verb, sing. present, non-3d Eg. 'take'
31	VBZ	verb, 3rd person sing. present Eg. 'takes'
32	WDT	wh-determiner Eg. 'which'
33	WP	wh-pronoun Eg. 'who', 'what'
34	WP\$	possessive wh-pronoun Eg. 'whose'
35	WRB	wh-abverb Eg. 'where', 'when'
36	QF	quantifier, Eg. 'lot', 'less'
37	VM	main verb
38	PSP	postposition
39	DEM	demonstrativ

The algorithm for the selection of key terms using POS tagger is given is algorithm

3

**Result:** Key Terms selected from the text using POS Tagging Method

```
1 specificLanguageText ← Load Specific Text Corpus into memory;  
2 tokenizedAnnotatedText ← posTag(specificLanguageText);  
3 candidateTerms ← Empty Set;  
4 foreach token in tokenizedAnnotatedText do  
5   | if getPOS(token) begins with 'NN' then  
6   |   | candidateTerms ← append(token);  
7   | else  
8   |   | Discard Token;  
9   | end  
10 end
```

**Algorithm 3:** Algorithm to select key terms using POS Tagging Method

As per algorithm 3, the input text is annotated using a POS tagger. The annotated text is taken one by one and if the POS tag is of 'Noun' type, i.e., it start with the POS tag 'NN' then, the token is added to the candidate terms, else it is discarded. The design specification for the complete single term extraction including the POS tagger option is given in figure 3.7

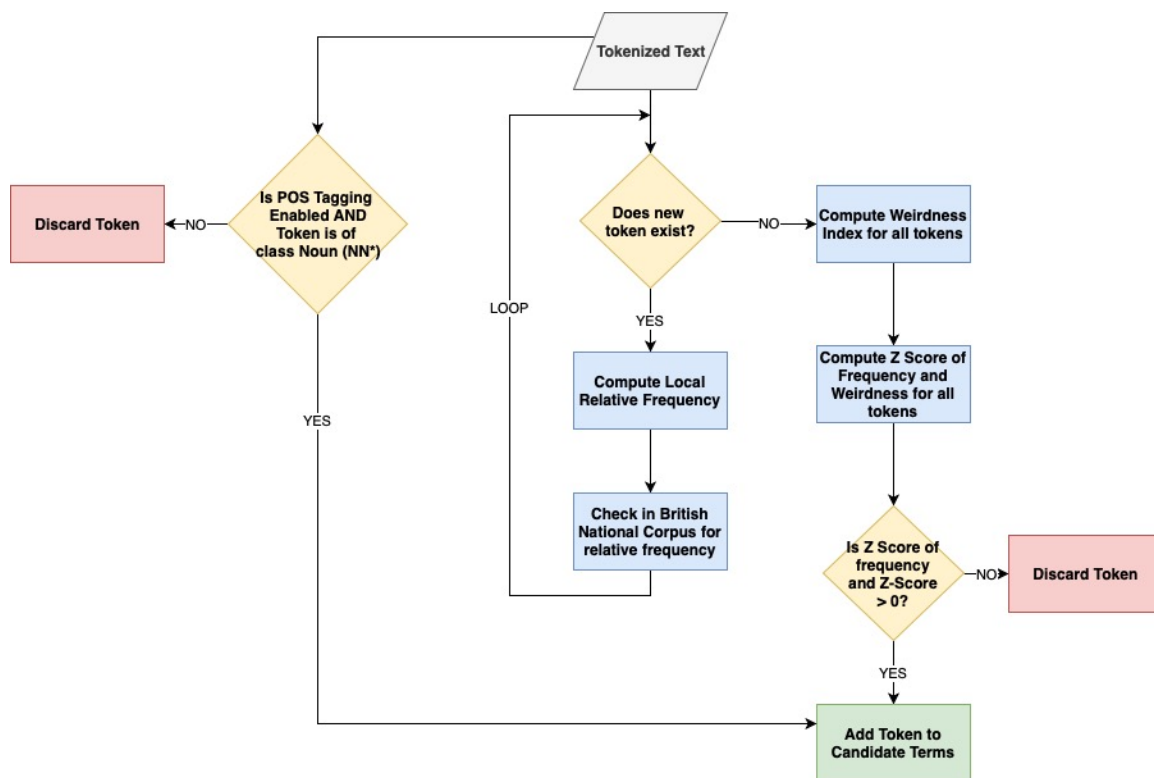


Figure 3.7: Design Specification of the selection of key terms including POS Tagging represented using UML Flow-chart

### Collocation Analysis

Collocates are those words that co-occur more often than by chance. Collocates corresponds to arbitrary word usages in the text and natural language text is filled with collocates [46]. Collocates can be computed manually by taking each key term identified and looking at the words that co-occur on the left and right of it. However, it is a human-intensive process and cannot be incorporated into an automated algorithm. Hence, there exists a need to determine the collocations automatically.

Collocations increase the efficiency of the summarization algorithm since they identify compound words that co-occur in the text. These compound words can be used to establish additional links between sentences thereby enabling the summarization algorithm to better understand closely related sentences.

For automated collocation identification, this work makes use of the statistical

measures suggested by Smadja [46]. The three important statistical measures that are suggested by Smadja to retrieve collocations are *spread*, *strength* and *distance*.

To perform the computations, the first step is to create a frequency distribution table as shown in figure 1, where for each candidate word  $W^k$ , the frequency of  $n$  terms  $W^i$  in the left and right side of the word is calculated. A sample table from the medical domain where the frequencies are calculated for the 5 open-class words to the left and right of the candidate terms is given in table 3.2.

Table 3.2: Table showing the frequency calculation for candidate terms (W) and other open class words (Wi) for 5 words to the left (L1 to L5) and 5 words to the right (R1 to R5)

W	Wi	Total Fre- quency	L5	L4	L3	L2	L1	R1	R2	R3	R4	R5
bleeding	acute	38	0	0	20	12	1	0	1	1	1	2
bleeding	upper	35	2	1	0	21	0	1	4	2	3	1
upper	bleeding	35	1	3	2	4	1	0	21	0	1	2
upper	endoscopy	33	1	0	0	1	0	29	0	0	0	2
endoscopy	upper	33	2	0	0	0	29	0	1	0	0	1
bowel	small	25	0	0	0	0	25	0	0	0	0	0
capsule	endoscopy	23	0	0	0	0	0	23	0	0	0	0
endoscopy	capsule	23	0	0	0	0	23	0	0	0	0	0
patients	bleeding	21	0	0	0	1	0	0	0	2	6	12
bleeding	patients	21	12	6	2	0	0	0	1	0	0	0
bleeding	lower	21	0	0	1	17	0	0	0	1	1	1
obscure	bleeding	20	0	0	0	0	1	1	15	2	0	1
bleeding	obscure	20	1	0	2	15	1	1	0	0	0	0
upper	colonoscopy	18	0	0	1	2	0	0	0	14	0	1
colonoscopy	upper	18	1	0	14	0	0	0	2	1	0	0
endoscopy	colonoscopy	17	0	0	2	0	0	0	14	0	1	0
colonoscopy	endoscopy	17	0	1	0	14	0	0	0	2	0	0

catheter	angiography	16	0	0	0	3	1	11	0	1	0	0
upper	acute	16	3	1	0	0	11	0	0	0	0	1

Once such a table is created, then the next step is to calculate the *strength* for each item in the table. The *strength* of a collocate is computed by the equation 3.5

$$strength(w^k, w^i) = \frac{f_i - \mu}{\sigma} \quad (3.5)$$

where

$f_i$  = Sum of the frequency of  $W_i$  among the  $n$  neighbourhood

$\mu$  = Average frequency of the words that appear in the  $n$  neighbourhood of  $W^k$

$\sigma$  = Standard Deviation of the words that appear in the  $n$  neighbourhood of  $W^k$

*ST* can be defined as the *Strength Threshold* which is threshold for a collocate ( $W^k, W^i$ ) to be considered as a candidate.

The next measure for selecting a candidate collocate is the distribution of the frequency of the collocate or its *spread*. The *spread* metric enables to statistically determine whether a collocate ( $W^k, W^i$ ) has a sharp increase in the histogram analysis indicating that its co-occurrence in the given position is not random but a deliberate use by the author of the text or is a compound word that is part of the language. The spread for a collocate ( $W^k, W^i$ ) is defined by equation 3.6

$$spread(w^k, w^i) = \frac{\sum_{j=-n}^n (f_j^i - \bar{f}_i)^2}{2 * n} \quad (3.6)$$

where

$f_j^i$  = Frequency of  $W^i$  in position  $j$

$\bar{f}_i$  = Average frequency of  $W^i$  that appear in the  $n$  neighbourhood of  $W^k$

$n$  = number of neighbourhood. Normally  $n$  is set to 5 and  $n \neq 0$

*ST* can be defined as the *Spread Threshold* which is threshold for a collocate ( $W^k, W^i$ ) to be considered as a candidate.

The final measure for the determination of a collocate candidate is the *distance*. The *distance* is for a collocate ( $W^k, W^i$ ) is calculated using the equation 3.7

$$distance(w^k, w^i) = abs(0 - index(max(W^i))) \quad (3.7)$$

where

$index(max(W^i))$  = Distance of the word  $W^i$  with respect to the word  $W^k$

Based on the three measures of *strength*, *spread* and *distance*, the collocate candidates can be selection using the criteria given in equation 3.8

$$\begin{aligned} strength(w^k, w^i) &> ST \\ spread(w^k, w^i) &> SP \\ distance &= 1 \end{aligned} \quad (3.8)$$

where

ST = *Strength Threshold* specified by the user, the default is 1

SP = *Spread Threshold* specified by the user, the default is 5

Using equation 3.8 as the deciding criteria for the selection of collocate candidates, the algorithm for collocate selection is shown in algorithm 5.



**Result:** N Neighbour frequency for each key term

```
1 specificLanguageText ← Load Specific Text Corpus into memory;  
2 sentenceSplit ← sentenceSplit(specificLanguageText);  
3 keyTerms ← CandidateTerms;  
4 frequencyMap ← Empty Map;  
5 foreach sentence in sentenceSplit do  
6   foreach term in keyTerms do  
7     if frequencyFrame IS NULL in  
8       frequencyMap[term][frequencyFrame] then  
9         frequencyMap[term][frequencyFrame] ← frequencyFrame;  
10      end  
11     if term in sentence then  
12       tokenizedSentence ← tokenize(sentence) foreach token in  
13         tokenizedSentence do  
14           if token is OPEN-CLASS then  
15             frequencyMap[term][frequencyFrame] ←  
16               computeOrUpdateFrequency(token);  
17           else  
18             Discard Token;  
19           end  
20         end  
21     end  
22 end
```

**Algorithm 4:** Algorithm to compute the frequency of N neighbours for each key term

**Result:** Collocate Candidates from the given text

```
1 specificLanguageText ← Load Specific Text Corpus into memory;
2 sentenceSplit ← sentenceSplit(specificLanguageText);
3 keyTerms ← CandidateTerms;
4 frequencyMap ← ExecuteFrequencyAnalysis (see 4);
5 frequencyFrame ← EmptyArray[n];
6 candidateCollocates ← EmptySet;
7 strengthThreshold = 1;
8 spreadThreshold = 5;
9 distanceThreshold = 1;
10 foreach term in frequencyMap do
11   | frequencyMap[term][strength] ←
12     | computeStrength(frequencyMap[term][frequencyFrame]);
13   | frequencyMap[term][spread] ←
14     | computeSpread(frequencyMap[term][frequencyFrame]);
15   | frequencyMap[term][distance] ←
16     | computeDistance(frequencyMap[term][frequencyFrame]);
17 end
18 foreach term in frequencyMap do
19   | strength ← frequencyMap[term][strength]; spread ←
20     | frequencyMap[term][spread];
21   | distance ← frequencyMap[term][distance];
22   | if strength  $\geq$  strengthThreshold AND spread  $\geq$  spreadThreshold AND
23     | distance  $\leq$  distanceThreshold then
24     | | candidateCollocates ← append(term)
25     | else
26     | | Discard term;
27     | end
28 end
```

**Algorithm 5:** Algorithm to select candidate collocates using the statistical measures Strength, Spread and Distance

The collocates are computed as per the algorithm 5. The first step of the algorithm

is to compute the  $n$  neighborhood frequency for each key term for each of the open-class word in all the sentences of the  $CORPUS_S$ . This is done by executing the algorithm 1. Once the frequency map for each of the key terms is available, then the next step is to calculate the *strength*, *spread* and *distance* for each of the collocate ( $W^k, W^i$ ). After the computation of the *strength*, *spread* and *distance*, then each collocate is examined to check if it meets the criteria for candidate collocates as defined in equation 3.8 and those collocates which pass the criteria are selected as candidate collocates. The design specification of the candidate collocate selection process is represented as a UML flowchart in figure 3.8.

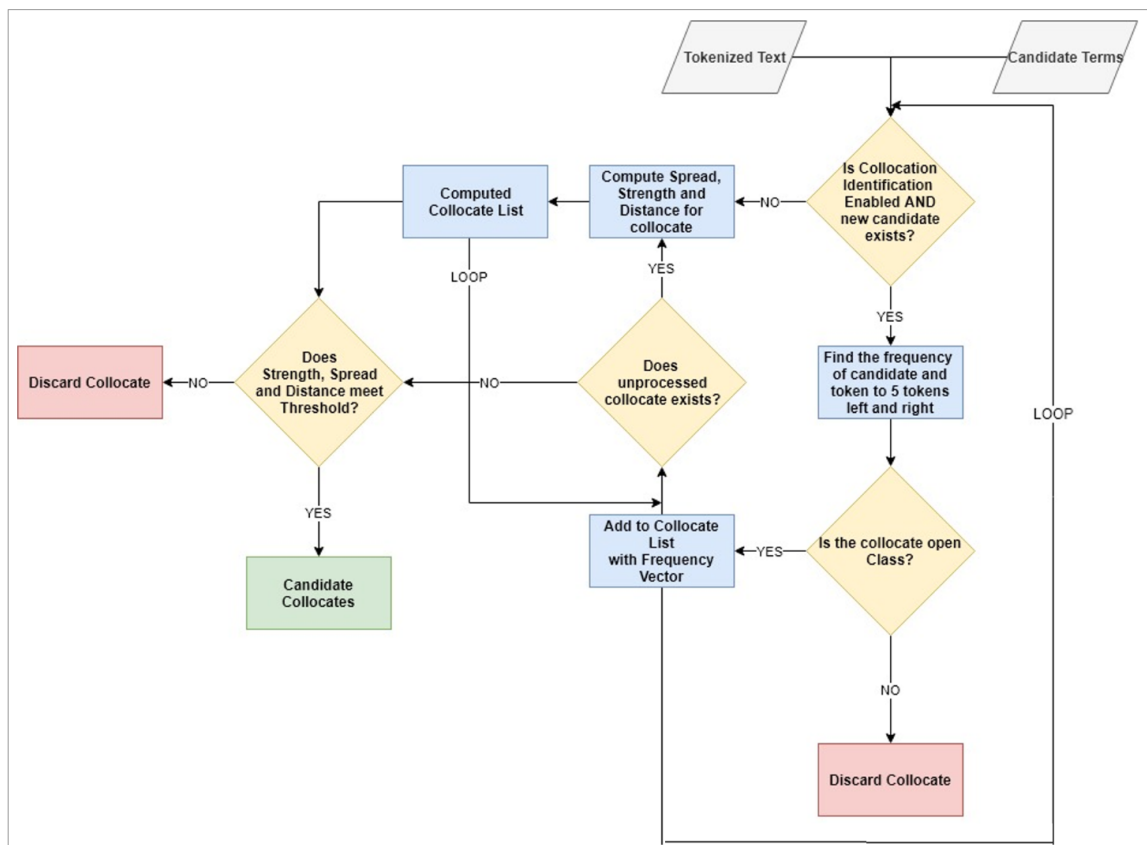
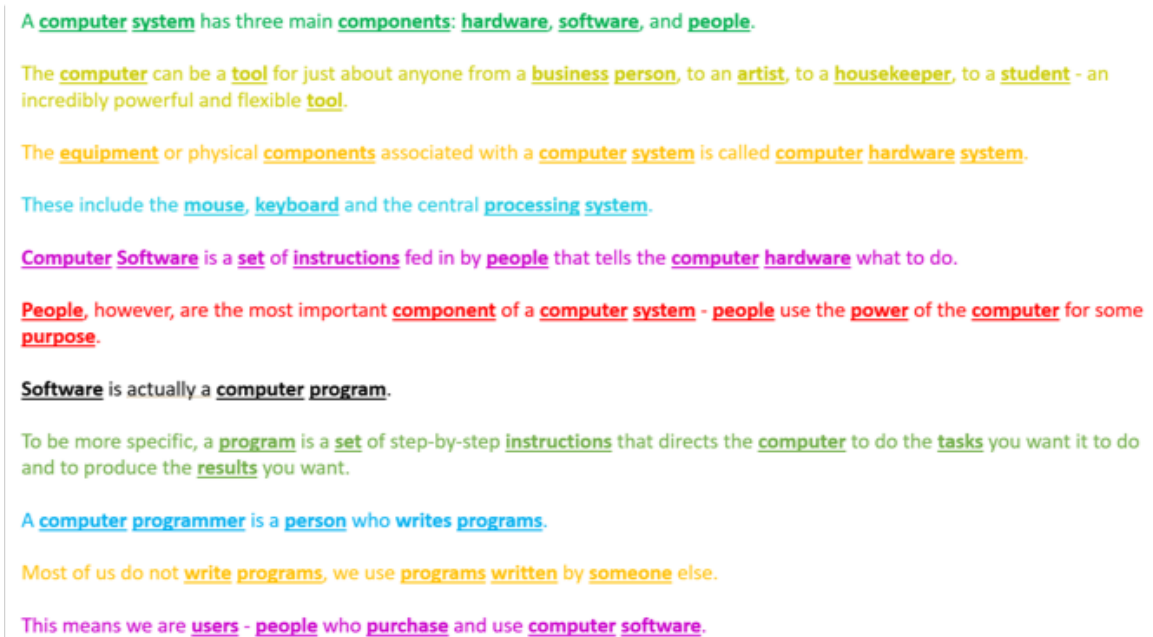


Figure 3.8: Design Specification of the collocate candidate selection represented using UML flowchart

### 3.3.2 Sentence Linking and Link Matrix

Once the key term identification is complete, logically the key terms annotated in each sentence. The logical view of the sentences from the sample text shown in figure 3.2 annotated with the identified key terms can be seen in figure 3.9. An configurable improvisation mechanism using the stemming of the key terms using the *Porter2 stemming algorithm* [47] is included in the sentence linking method.



A computer system has three main components: hardware, software, and people.

The computer can be a tool for just about anyone from a business person, to an artist, to a housekeeper, to a student - an incredibly powerful and flexible tool.

The equipment or physical components associated with a computer system is called computer hardware system.

These include the mouse, keyboard and the central processing system.

Computer Software is a set of instructions fed in by people that tells the computer hardware what to do.

People, however, are the most important component of a computer system - people use the power of the computer for some purpose.

Software is actually a computer program.

To be more specific, a program is a set of step-by-step instructions that directs the computer to do the tasks you want it to do and to produce the results you want.

A computer programmer is a person who writes programs.

Most of us do not write programs, we use programs written by someone else.

This means we are users - people who purchase and use computer software.

Figure 3.9: Logical view of the sentences annotated with the identified key terms

The next step of the algorithm is the sentence linking process. The process enables to find the links between the sentences based on the key terms identified as explained in section 3.3.1.

The process is executed by taking each key term identified and annotated in a sentence and looking for that key term in all the other sentences. If the same key term is found in another sentence, then a link is created between the origin sentence ( $S_o$ ) and the target sentence ( $S_t$ ) and the link is marked as an out-link in the origin sentence ( $S_o$ ) and in-link in the target sentence ( $S_t$ ). The algorithm for the process of sentence linking is given in algorithm 6

**Result:** Link Matrix

```
1 sentenceSplit ← keyTermsMarkedSentence;  
2 foreach  $S_o$  in sentenceSplit do  
3   | foreach  $S_t$  in sentenceSplit do  
4   |   | if  $S_t \neq S_o$  then  
5   |   |   | foreach term in  $S_o[\textit{terms}]$  do  
6   |   |   |   | if term in  $S_t[\textit{terms}]$  then  
7   |   |   |   |   |  $S_o[\textit{out}] \leftarrow \textit{index}(S_t)$   $S_t[\textit{in}] \leftarrow \textit{index}(S_o)$   
8   |   |   |   |   | end  
9   |   |   |   | end  
10  |   |   | end  
11  |   | end  
12 end
```

**Algorithm 6:** Algorithm to link key term annotated sentences in a text

After the execution of this algorithm, the link between each sentence of the text will be established. Considering the example in 3.2, and considering only one key term, say 'computer' figure 3.10.

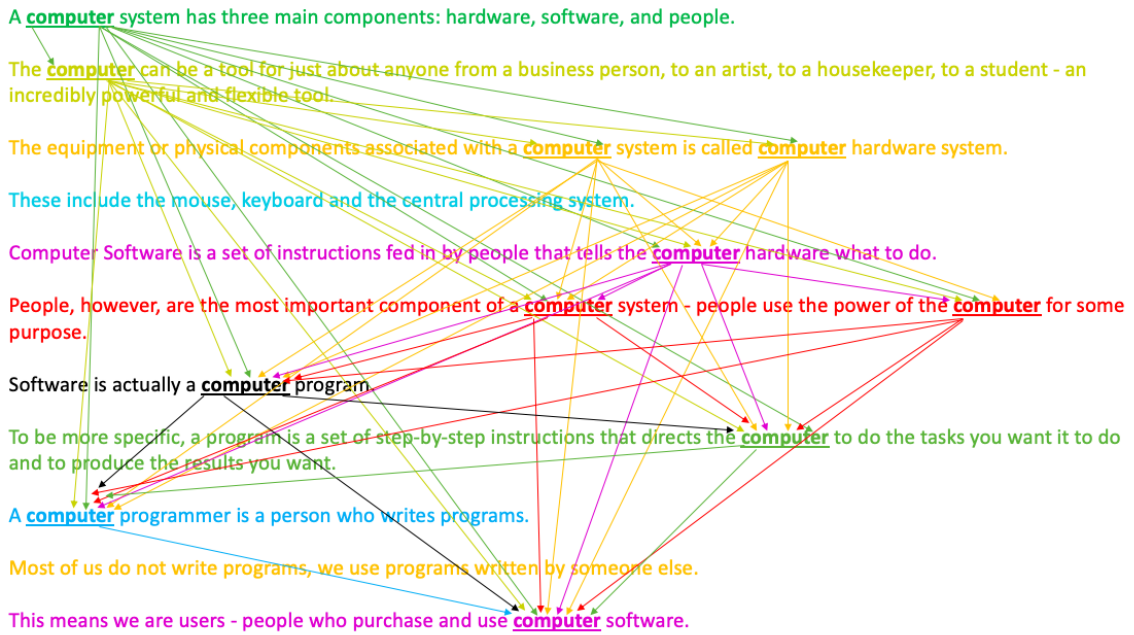


Figure 3.10: Network among various sentences based on key terms for incoming, outgoing and intermediate sentence analysis

The output of the algorithm 6 will be a  $n \times n$  matrix with the count of links from one sentence to the other that is symmetric over the leading diagonal. The link matrix for the sample text shown in figure 3.2 is shown in 3.11

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
s1	0	1	6	1	3	6	1	1	1	0	3
s2	1	0	2	0	1	2	1	1	2	0	1
s3	6	2	0	1	2	4	1	1	1	0	1
s4	1	0	1	0	0	1	0	0	0	0	0
s5	3	1	2	0	0	4	2	3	1	0	2
s6	6	2	4	1	4	0	1	1	1	0	2
s7	1	1	1	0	2	1	0	2	2	2	1
s8	1	1	1	0	3	1	2	0	2	2	1
s9	1	2	1	0	1	1	2	2	0	2	1
s10	0	0	0	0	0	0	2	2	2	0	0
s11	3	1	1	0	2	2	1	1	1	0	0

Figure 3.11: The link matrix showing the count of links between sentences

### 3.3.3 Sentence Bonding and Bond Matrix

The process of bonding is nothing but finding out the sentences that are closely related to each other based on the number of links between the text. To identify closely related sentences, the threshold value called as *bond<sub>s</sub>strength* is introduced.

The threshold value *bond<sub>s</sub>strength* can be defined as the minimum number of links that needs to exist between two sentences in order to declare the sentences as bonded. Bonding of sentences enable to filter out the marginal or non-valuable sentences from the text and enable the creation of a better summary.

The algorithm for the bond creation process is given in algorithm 7. As per the algorithm, each sentence is traversed and its in-links and out-links are examined. If there are more links than the *bond<sub>s</sub>strength* to a sentence, then based on whether it is an in-link or an out-link, an in-bond or out-bond is added from the source sentence ( $S_o$ ) and target sentence ( $S_t$ ).

**Result:** Bond Matrix

```

1 sentenceSplit ← linkedSentences;
2 bondStrength ← userDefinedBondStrength;
3 foreach  $S_o$  in sentenceSplit do
4   | inLinks ←  $S_o$ [inLinks] outLinks ←  $S_o$ [outLinks] foreach inLink in
5   |   | inLinks do
6   |   |   | if inLink[count]  $\geq$  bondStrength then
7   |   |   |   |  $S_o$ [inBonds] ← inLink[index];
8   |   |   |   | end
9   |   |   | end
10  |   | foreach outLink in outLinks do
11  |   |   | if outLink[count]  $\geq$  bondStrength then
12  |   |   |   |  $S_o$ [outBonds] ← outLink[index];
13  |   |   |   | end
14  |   |   | end
15  | end

```

**Algorithm 7:** Algorithm to establish bonds between sentences

The links that will be selected and determined as bonds based on the link ma-

trix shown in figure 3.11 is highlighted with and shown in the figure 3.12 when the  $bond_s trength$  is set as 2. The change in the bond matrix when the  $bond_s trength$  is changed to 3 is shown in the figure 3.13. From the figures 3.12 and 3.13 it can be seen that the  $bond_s trength$  is an important parameter in determining the bonds between sentences. Since the summary generated is directly dependent on the bonds between sentences, it is important to tune the  $bond_s trength$  to an optimal value for effective summary generation.

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
s1	0	1	6	1	3	6	1	1	1	0	3
s2	1	0	2	0	1	2	1	1	2	0	1
s3	6	2	0	1	2	4	1	1	1	0	1
s4	1	0	1	0	0	1	0	0	0	0	0
s5	3	1	2	0	0	4	2	3	1	0	2
s6	6	2	4	1	4	0	1	1	1	0	2
s7	1	1	1	0	2	1	0	2	2	2	1
s8	1	1	1	0	3	1	2	0	2	2	1
s9	1	2	1	0	1	1	2	2	0	2	1
s10	0	0	0	0	0	0	2	2	2	0	0
s11	3	1	1	0	2	2	1	1	1	0	0

Figure 3.12: The Bond matrix showing the detection of bonds from link matrix when the bond strength is set as 2



	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11
s1	0	1	6	1	3	6	1	1	1	0	3
s2	1	0	2	0	1	2	1	1	2	0	1
s3	6	2	0	1	2	4	1	1	1	0	1
s4	1	0	1	0	0	1	0	0	0	0	0
s5	3	1	2	0	0	4	2	3	1	0	2
s6	6	2	4	1	4	0	1	1	1	0	2
s7	1	1	1	0	2	1	0	2	2	2	1
s8	1	1	1	0	3	1	2	0	2	2	1
s9	1	2	1	0	1	1	2	2	0	2	1
s10	0	0	0	0	0	0	2	2	2	0	0
s11	3	1	1	0	2	2	1	1	1	0	0

Figure 3.13: The Bond matrix showing the detection of bonds from link matrix when the bond strength is set as 2

### 3.3.4 Sentence Categorization using in-out bond ratio

The penultimate stage before the creation of the summary is the categorization of links as *OPENING*, *CLOSING*, *MIDDLE* and *MARGINAL*. This categorization enables the Intelligent Text Summarization algorithm to produced effective summaries by identifying not just the importance sentences, but also the logical position of the sentence in the summary.

*OPENING* sentences are ones that introduce the core topics of the text. *MIDDLE* sentences generally elaborate on the topic and the *CLOSING* sentences conclude the topic. The *MARGINAL* sentences are those sentences that are not important to the text and when removed do not lose information. Hence all sentences which are not bonded are classified as *MARGINAL*

The in-out bond ratio is calculated as per the formula given in equation 3.9.

$$BondRatio(BR) = \frac{count(S_{in}^i)}{count(S_{out}^i)} \quad (3.9)$$

where

$count(S_{in}^i)$  = Count of incoming bonds for the sentence  $i$

$count(S_{out}^i)$  = Count of outgoing bonds for the sentence  $i$

Based on the bond ratio ( $BR$ ), the links are classified as per the equation 3.10.

$$\begin{aligned}
 OPENING &\leftarrow BR < 0.5 \\
 MIDDLE &\leftarrow BR \geq 0.5 \text{ AND } BR \leq 1.5 \\
 CLOSING &\leftarrow BR > 1.5
 \end{aligned} \tag{3.10}$$

The algorithm for sentence classification based on the bond ratio is given in algorithm 8. The algorithm works by first computing the bond ratio for each sentence based on equation 3.9 and then classifying the sentence based on the equation 3.10.

**Result:** Categorized Sentences

```

1 sentenceSplit  $\leftarrow$  bondedSentences;
2 foreach  $S_o$  in sentenceSplit do
3   | inBonds  $\leftarrow$   $S_o$ [inBonds];
4   | outBonds  $\leftarrow$   $S_o$ [outBonds];
5   | bondRatio = count(inBonds) / count(outBonds);
6   | if bondRatio  $\leq$  0.5 then
7     |    $S_o$ [type]  $\leftarrow$  OPENING ;
8     |   continue;
9   | end
10  | if bondRatio  $\geq$  0.5 AND bondRatio  $\leq$  1.5 then
11  |    $S_o$ [type]  $\leftarrow$  MIDDLE ;
12  |   continue;
13  | end
14  | if bondRatio  $>$  1.5 then
15  |    $S_o$ [type]  $\leftarrow$  CLOSING ;
16  |   continue;
17  | end
18 end

```

**Algorithm 8:** Algorithm to categorize the bonded sentences based on bond ratio

### **3.3.5 Summary Generation**

The final step of the algorithm is summary generation. The summary generation is done by grouping all the opening sentences, middle sentences and closing sentences based on the categorization done using the bond ratio. This enables the summary to be logically relevant and organized based on the flow of the topic. Within each group, the sentences are placed in the same order as they appeared in the input text.

The algorithm for summary generation is given in algorithm 9.

**Result:** Summary of the input Text

```
1 sentenceSplit ← categorizedSentences;
2 openingSentences ← EmptyArray;
3 closingSentences ← EmptyArray;
4 middleSentences ← EmptyArray;
5 summary ← EmptyString;
6 foreach  $S_o$  in sentenceSplit do
7   if  $S_o$ [type] == OPENING then
8     | openingSentences ← append( $S_o$ [text]);
9     | continue;
10  end
11  if  $S_o$ [type] == CLOSING then
12    | closingSentences ← append( $S_o$ [text]);
13    | continue;
14  end
15  if  $S_o$ [type] == MIDDLE then
16    | middleSentences ← append( $S_o$ [text]);
17    | continue;
18  end
19  foreach sentence in openingSentences do
20    | summary ← summary + sentence;
21  end
22  summary ← summary + NEWLINE;
23  foreach sentence in middleSentences do
24    | summary ← summary + sentence;
25  end
26  summary ← summary + NEWLINE;
27  foreach sentence in closingSentences do
28    | summary ← summary + sentence;
29  end
30  summary ← summary + NEWLINE;
31 end
```

**Algorithm 9:** Algorithm to produce summary form the bond ratio based categorized sentences

## 3.4 Evaluation Method

When it comes to automatic text summarization, it is important to verify that the summaries that are produced are meaningful, does not lose the information from the main text and is easy for people to read and understand. However, the evaluation of the summaries produced by automatic text summarization tools is human-intensive. Moreover, the evaluation is subjective and varies from human to human. Even if the strategy is to compare a human-created summary to an automated summary, the human-created summaries vary based on the technical interest, language skill-set and domain expertise of the author of the summary. Hence, there exists a need to perform an automatic evaluation of the summaries produced in a statistically significant way especially when it involves the evaluation of summaries produced for a large corpus of text. Subjective human evaluation can be used for cross-checking the automatic summarization method and can be done in a minimal and controlled manner.

Hence to perform an automatic evaluation of the summaries produced, three methods are used in combination in this work.

1. Readability Index (see section 3.4.1)
2. Correlation of the Relative Frequency Method (see section 3.4.2)
3. Cumulative Relative Frequency Comparison of top open-class words (see section 3.4.3)

### 3.4.1 Readability Index

Readability index is an automatic method of collecting data from a text in order to gauge its understandability and ease of reading. There are several readability indices available such as Gunning Fox Index, Automated Readability score, Smog Index, etc. However, one of the most widely used and validated readability index methods is Flesch Kinkaid Reading Ease Formula. The reading ease formula is used for identifying the ease of reading a given document. The formula is based on the number of sentences, words, and syllables that exist in the document. More specifically it depends on the ratio of the number of words to the number of sentences and the number of syllables to the number of words. It is computationally easy to use method for evaluating

the readability of a text document [48]. The Flesch Kinkaid Reading Ease formula is defined by the equation 3.11.

$$ReadingEase = 206.835 - (1.015 * TW/TSe) - (84.6 * TSy/TW) \quad (3.11)$$

where

TW = Total number of words in the text

TSe = Total number of sentences in the text

TSy = Total number of syllables in the text

The output of the reading ease formula is a number in the range 0 to 100. The interpretation of the score is given in table 3.3

Table 3.3: Flesch Kinkaid Reading Ease interpretation reckoner

Score	School level	Notes
100.00 to 90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0 to 80.0	6th grade	Easy to read. Conversational English for consumers.
80.0 to 70.0	7th grade	Fairly easy to read.
70.0 to 60.0	8th to 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0 to 50.0	10th to 12th grade	Fairly difficult to read.
50.0 to 30.0	College	Difficult to read.
30.0 to 0.0	College graduate	Very difficult to read. Best understood by university graduates.

The interpretation of table 3.3 is that, if the reading ease score of a text is in the range of 90 to 100, then a person who has knowledge of English in the level that is taught in the 5th grade can read the text. In other words, the closer the score is to 100,

the easier is it to read the document. Normally scholarly articles contain a reading ease score less than 50 and a majority of them less than 30.

In this work, the Flesch Kinkaid Reading Ease is calculated for the input text and the summary that is produced. The result is evaluated by comparing the reading ease score for the input text and the summary.

### **3.4.2 Correlation of Relative Frequency**

In this method of automatic evaluation, the relative frequency for each word is calculated the input text and their corresponding relative frequency in the summary that is produced. The relative frequency is calculated for both the open and closed-class words. Then the correlation of the vectors of relative frequencies in the input text and summary text is determined. Since the summary should be a true representation of the input text, the correlation of the relative frequency of words in the summary with that relative frequency of words in the input text should be very high. A low correlation will indicate that there is information loss in the summary that is produced.

The algorithm for calculation of the correlation between the relative frequency in the given in [algorithm 10](#)

**Result:** Correlation between the relative frequency of words in input text and summary text

```
1 inputTokens ← tokenizedInputText;  
2 summaryTokens ← tokenizedSummaryText;  
3 initialize(inputWordCount, summaryWordCount, inputTokenMap,  
   summaryTokenMap, inputTextVector, summaryTextVector, correlation);  
4 foreach token in inputTokens do  
5   | inputWordCount ← inputWordCount + 1;  
6   | if token in inputTokenMap then  
7     | inputTokenMap[token] ← inputTokenMap[token] + 1;  
8   | else  
9     | inputTokenMap[token] ← 1;  
10  | end  
11 end  
12 foreach token in inputTokenMap do  
13   | inputTextVector ← inputTokenMap[token] / inputWordCount;  
14 end  
15 foreach token in summaryTokens do  
16   | summaryWordCount ← summaryWordCount + 1;  
17   | if token in summaryTokenMap then  
18     | summaryTokenMap[token] ← summaryTokenMap[token] + 1;  
19   | else  
20     | summaryTokenMap[token] ← 1;  
21   | end  
22 end  
23 foreach token in summaryTokenMap do  
24   | summaryTextVector ← summaryTokenMap[token] /  
   | summaryWordCount;  
25 end  
26 correlation ← computeCorrelation(inputTextVector, summaryTextVector)
```

**Algorithm 10:** Algorithm for evaluation of summary by computing the correlation between the relative frequency of word vectors in input text and summary text



The output of algorithm 10 gives the correlation of the relative frequencies between the word vectors of input text and summary text which is a statistical measure of determining whether the summary text contains the information that is present in the input text.

### 3.4.3 Cumulative Relative Frequency Comparison of top open-class words

When a summary for a text is generated, the quality of the summary is based on whether then summary is understandable by the reader and whether the summary contains all the relevant information contained in the input text. The measure of whether the reader can understand the summary is a conceptual measure. However, the conceptual measure is difficult to be computed automatically and needs subjective human evaluation. On the other hand, the understandability of the summary generated can be analyzed linguistically. Linguistic theories include semantic studies which again is computationally difficult, however, the cohesion studies enable the computation of readability and understanding using simple mechanical computation and can be equated to the conceptual measure of understandability of the summary.

The idea is that in a scientific text authors report, elaborate and aggregate topics. To do this, there is a theme that is followed and the important topics are repeated throughout the text by the usage of topics and terms. The sentences in the text are linked using lexical linkages and semantics of the language. The semantics of the language is difficult to be computed, however, the lexical linkages can be analyzed using a bag of words model. The lexical linkages analysis work especially good on *language for special purposes*. Language for special purposes (LSP) make use of the special or domain-specific words which are introduced by the authors and have a special meaning in the domain. These words that are specific to the domain is called *open-class words* and form a small slither of the LSP and this concept is called *Neologism* [49]. The open-class words enable the lexical cohesion in the text which is indirectly related to the semantics of the text.

Based on the understanding of Neologism and its part in LSP, the goodness of a summary can be evaluated using the cumulative relative frequency of the open-class words in the input text and summary text. Considering a scientific text which is

jargon-rich, the top 100 words contribute to the meaning of the text. By analyzing the cumulative relative frequency of the top meaning contributing words in the input text, and analyzing the cumulative relative frequency of the same 100 words in the summary, it is possible to determine the goodness of the summary statistically verifiable manner. The cumulative relative frequency of the top 100 words in the text can be calculated using the algorithm 11.

**Result:** Cumulative Relative Frequency Vector of the given text

```

1 wordFrequencyMap ← computeFrequencyMap(text);
2 wordCount ← computeTotal(wordFrequencyMap);
3 counter ← 0;
4 initialize(cumulativeFrequencyVector);
5 wordFrequencyMap ← sort(wordFrequencyMap);
6 while wordFrequencyMap has token do
7   | if token is OPEN-CLASS then
8   |   | counter ← counter + 1;
9   |   | cumulativeFrequencyVector ← token[count] / wordCount;
10  | end
11  | if counter ≥ 100 then
12  |   | break;
13  | end
14 end

```

**Algorithm 11:** Algorithm to calculate the cumulative relative frequency of the top open-class words in a text

## 3.5 Conclusion

This chapter explains in details the various steps involved in the generation of a summary from an input scientific text. The various section elaborate in detail the different methods used for pre-processing the text (see section 3.2), and Intelligent Text Summarization (see section 3.3).

For each of the distinct methods, detailed algorithms and illustrations are provided to explain the concept with an easy to understand practical example of a sample text

shown in figure 3.2.

Though the algorithm performs better on large scientific texts, to complete the illustration of the method used for *Intelligent Text Summarization*, the output summary generated by the algorithm for the sample text (see figure 3.2) is given in figure 3.14.

A computer system has three main components: hardware, software, and people. The equipment or physical components associated with a computer system is called computer hardware system. Computer Software is a set of instructions fed in by people that tells the computer hardware what to do. People, however, are the most important component of a computer system - people use the power of the computer for some purpose.

Figure 3.14: The summary generated using the Intelligent Summarization Algorithm with the Sample Text as input

The evaluation for the goodness of the summary using various automated statistically verifiable methods that can be used for large corpus of text is explained in section 3.4.

The complete design specification of the 'Intelligent Text Summarization' algorithm is represented as a UML flowchart in figure 3.15

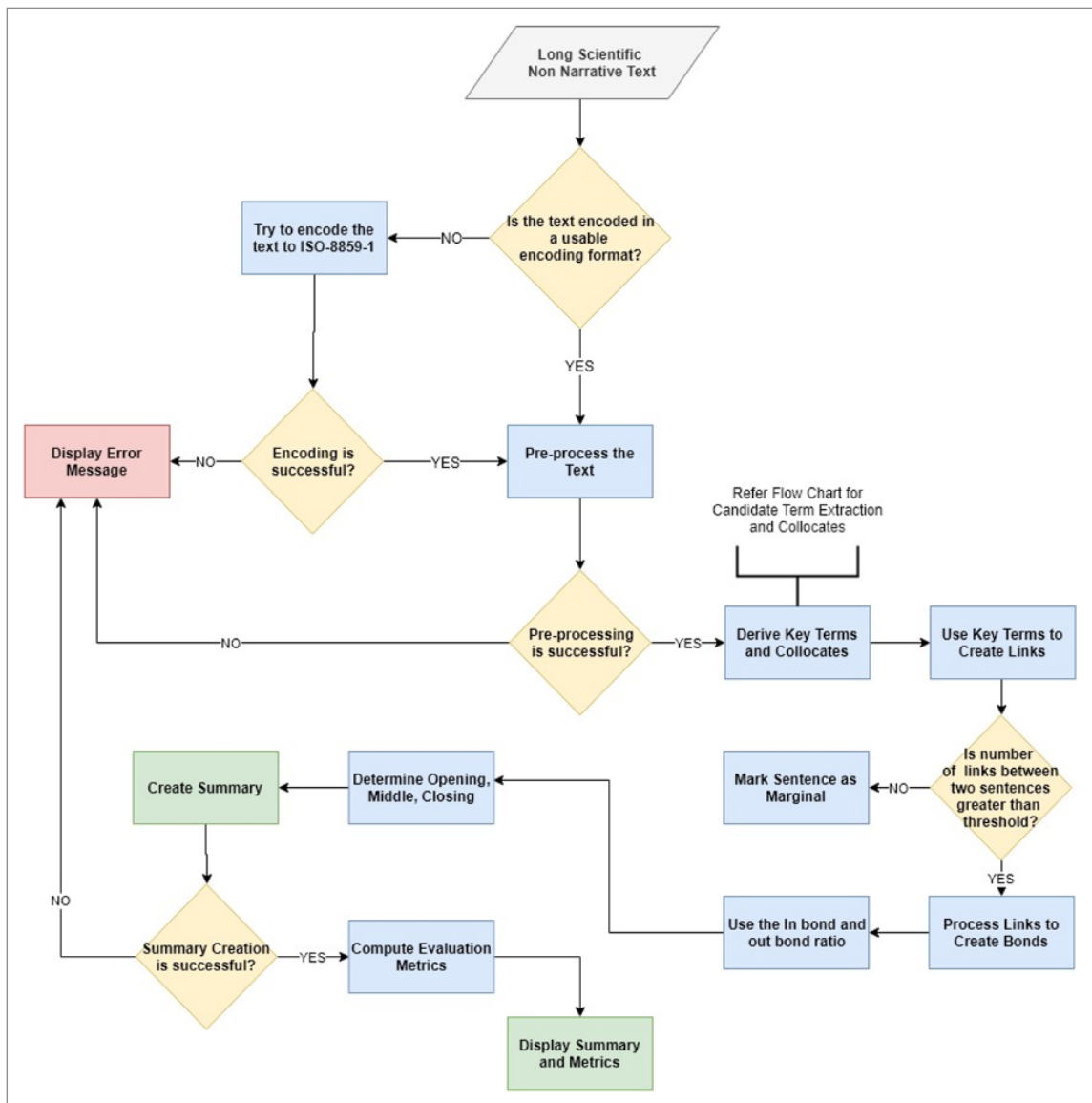


Figure 3.15: Design Specification of Intelligent Text Summarization Algorithm UML flowchart

The implementation of the *Intelligent Text Summarization* algorithm and controlled case studies to evaluate the implementation is explained in chapter 4.

# Chapter 4

## Implementation and Case Studies

### 4.1 Introduction

This chapter explains the implementation of the *Intelligent Text Summarization* algorithm for which the method was explained in chapter 3. The section provides information regarding the choice of dataset and the rationale behind the selection (see section 4.2). It also explains the technical specification of the system by explaining the architecture and the user interface design (see section 4.3). In the later part of this chapter, case studies in which controlled experimentation with instrumentation was done to evaluate the system is presented with the results obtained along with a discussion of the results obtained (see section 4.4). The final part of this chapter provides insights into the system performance on different scenarios and system load (see section 4.5). The security and privacy considerations with respect to the implementation is given in section 4.6. The chapter is concluded in the section 4.7

### 4.2 Dataset Selection

The dataset that is vital for testing and evaluation of the implementation of *Intelligent Text Summarization*. The dataset needs to be carefully chosen to cover a wide variety of scenarios so that the implementation can be subjected to different use cases and the algorithm can be evaluated well.

In this work, the primary focus is on the automated text summarization of large

scientific texts and since the research is in the field of computer science, one of the dataset chosen for evaluation is large scientific texts in the field of computer science. The works by *Alan Turing* is considered as one of the datasets from the computer science domain. Alan Turing is a renowned computer scientist and his contribution to the field such as compilers, cryptography, artificial intelligence, etc., is well respected. Since this work considers contributes to the field of artificial intelligence and the articles by Alan Turing are long texts in the scientific domain, the usage of the articles by him deems to be a well-considered choice. However, many of the articles by him are not in the digital format and hence three prominent articles that are digitized and available for public use is chosen as one of the datasets in this work.

The articles by Alan Turing used for evaluation in this work are as follows.

1. Computing Machinery and Intelligence [50]
2. On computable numbers, with an application to the Entscheidungsproblem [51]
3. The Chemical Basis of Mokphogenesis [52]

Apart from the works by Alan Turing, 7 other articles from the field of computer science are chosen to be a part of the dataset related to computer science. Some of them are *Steps towards Artificial Intelligence* [53], *Establishing Moore's law* [54] and *Origins of Domain Name System* [55]. The complete set of titles of the articles used for evaluation is provided in the appendix.

Apart from the dataset in the computer science domain, to test the practicability of the proposed algorithm in large scientific texts, a set of fifteen long scientific articles from the bio-medicine (gastroenterology) domain is chosen as another dataset.

Also, as experimentation, a set of fifty thousand tweets collected during the Russian River Flooding in the Sonoma County, California during March 2019 is used to explore whether the same algorithm can be used to identify the most important information present in a large cluster of tweets. The experiment was conducted to explore if the algorithm, the core of which is to identify important sentences in a text and order them logically, can be used in fields like *Social Media Analytics for Disaster Management* where large volumes of social media message have to understood and prioritized for emergency management organizations to take action to save life and property of the

citizens. This experiment is not a core part of this work, however, is used to show the applicability of the algorithm in other domains.

## **4.3 Technical Specification and System Design**

This section explains the system architecture of the implementation, the technical stack used and the rationale for the choice and the user interface design of the application that was developed to showcase the implementation of the Algorithm.

### **4.3.1 System Architecture**

The system is architected using a *service-oriented architecture* (SOA) principles in a client-server approach. The component diagram of the system architecture is given in figure 4.1. The system consists of two major sections - user interface application and the Intelligent Text Summarization server.

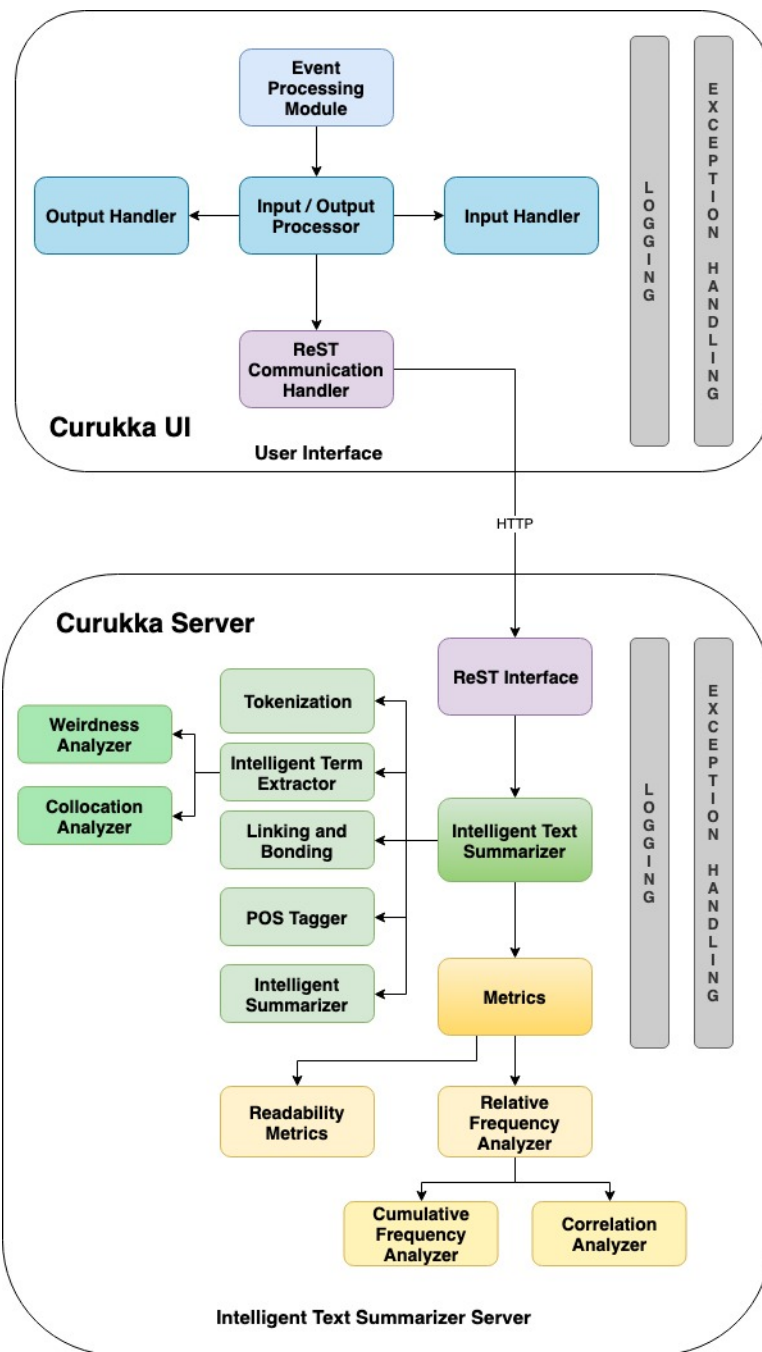


Figure 4.1: System Implementation Design of the Intelligent Text Summarization - Curukka System



## User Interface Application

The user-facing part of the system is the user interface application that is built as a standalone application that can be used on both *Microsoft Windows 10+* and *Apple Mac OSX 10.14+* operating systems. The user interface application consists of an *event processing module* that listens to user events and invokes the appropriate methods. This is the primary user interaction module of the user interface application and handles both mouse interactions and keyboard inputs. Another important component of the user interface application is the *input/output processing module*. The input/output processing module is responsible for the handling of file operations such as opening a file for summarization, writing the output of the application to the file, and rendering the output from the server on the user interface application. The user interface application contacts the server module over HTTP using ReST application programming interface (API). The rest communication component of the user interface application takes care of all the communication-related activities including the creation of server requests, processing the response from the server. Each component of the user interface application also takes care of exception handling, gracefully handling the errors and logging them for debugging in case of a need. The user interface application consists of 1681 lines of code.

## Intelligent Text Summarization Server

The server module of the system consists of three major components. The *ReST interface component* is responsible for receiving requests from the clients and sending the response back. The ReST interface component is multi-threaded and can handle multiple requests simultaneously. The number of requests that can be handled simultaneously depends on the hardware and software configuration of the system and can be configured by the user. The *Intelligent Text Summarizer Component* contains the implementation of the core algorithm. It is invoked by the ReST interface component to server user requests. The *Metrics component* of the server work along with the summarizer component. The metrics component provides all the instrumentation metrics required to evaluate the working of the system and also records the performance metrics. As in the user interface application, each component of the server application takes care of the exception handling and logging.

### 4.3.2 Technical Stack

The application is built using two major technical stacks. The user interface of the application is built using the JAVA programming language. The *Swing Windows Toolkit* (SWT) and the *Advanced Widgets Toolkit* (AWT) along with the *Windows builder framework* for JAVA programming language is used for developing the user interface. The application development is done using *Eclipse 2019-03* integrated application development environment with *JAVA Development Kit (JDK) version 1.8*. The user interface application was developed on *Apple OSX Mojave 10.14* operating system and the distributables are built to run on both *Microsoft Windows* and *Apple OSX* operating systems with *Java Runtime Environment (JRE) version 1.8* installed. The rationale behind the choice of a technical stack based on the JAVA programming language for the user interface application is because of the platform independency provided by JAVA programming language, the availability of Windows desktop application development toolkits and the excellent community support.

The technical stack for the server is based on the *Python programming language*. The ReST interface module is built using the *Python Flask framework*. The metrics module of the server component makes use of the popular number processing frameworks in the Python stack such as *numpy* and *pandas*. The choice of a Python based technical stack to implement the core algorithm and the server component is due to the native support for advanced text processing in the Python programming language. This is evident from the fact that most popular natural language processing systems like the Stanford NLP system are build using Python programming language. The server component is deployed as one command executable that can be run on any Linux based environment that has support for Python programming language version 3.7. The server module consists of 1585 lines of code.

The detailed instruction on the deployment and usage of the application is given in the appendix.

### 4.3.3 User Interface Design

This section explains the user interface design of the *Curukka* system. The user interface consists of primarily eight screen views, each of which are explained below with a screenshot.

## Configuration Panel

The configuration panel is the single point of all user configurations for the Curukka system. The screenshot of the configuration panel can be seen in figure 4.2. The bond strength threshold that should be used for the summarization process is available as an editable dropdown list in the configurations group. The default is set as '3', however, the user can choose the required bond strength from the dropdown list or enter a number in the range of 2 to 10. The configuration group also consists of a set of radio buttons which allows the user to choose whether the term extraction should be done using the only the weirdness index method or to use a hybrid approach of both weirdness index method and POS tagging method. The default setting is to use only the weirdness index method for term extraction.

The corpus selection configuration group enables the user to configure the reference corpus that should be used. The user has a choice between the Open American National Corpus (ANC) or the British National Corpus (BNC). The default option is set as BNC.

The powerups configuration enable the user to select whether to use collocation analysis for text summarization and enable or disable the use of stemming or terms before the analysis is performed.

The final configuration item is an experimental one which enables the user to indicate to the system that the text being summarized are tweets. When this option is selected the only difference is that when the summary output is rendered, instead of the producing paragraphs, the system arranges the tweet one per line.

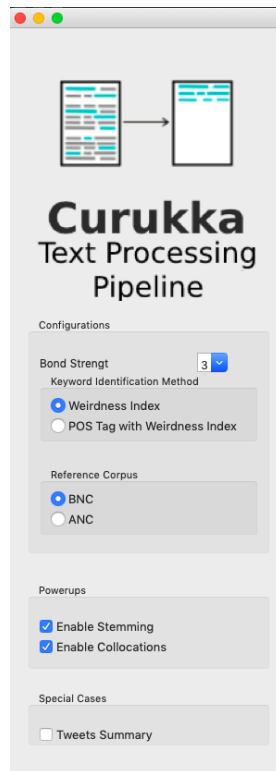


Figure 4.2: Screenshot of the configuration panel for the Curukka System

### Text Input Screen

The text input screen allows the user to enter the input text. The input can be added by typing directly into the screen, or by pasting the text into the screen from clipboard, or by the use of the menu bar to open a file (in '.txt' format). A screenshot of the text input screen is shown in figure 4.3

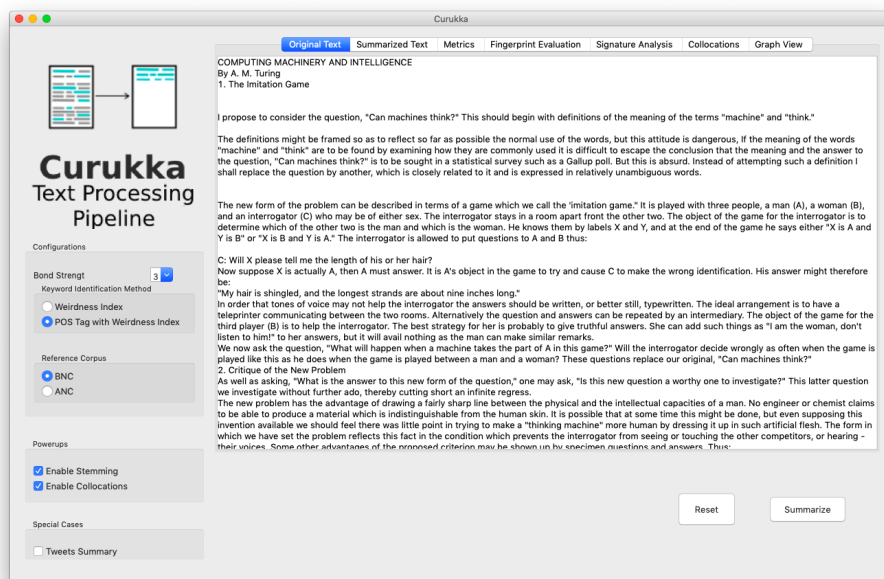


Figure 4.3: Screenshot of the Text Input Screen for the Curukka System

## Summary Screen

The summary screen is used to display the summary to the user. The summary screen does not allow the user to add or edit the text in it. Once the summary is received from the server, the application renders the summary on the summary screen and automatically switches the view to it. The summary rendered on the summary screen can be saved using the save menu option. A screenshot of the summary screen is shown in figure 4.4

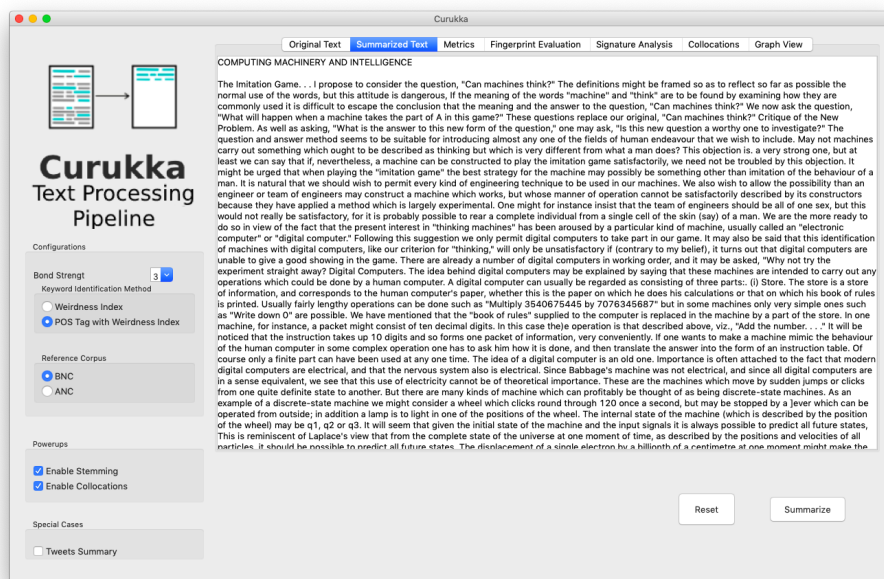


Figure 4.4: Screenshot of the Summary Screen for the Curukka System

## Metrics Screen

The metrics screen is used to show the important metrics of after the completion of the summarization. A screenshot of the metrics screen is shown in figure 4.5. The various metrics present on the metrics screen are as follows.

- Bond Strength*: The bond strength that was input by the user.
- Total Sentences in Full Text*: The total number of sentences that were present in the input text.
- Total Sentences in Summary Text*: The total number of sentences that are present in the summary text.
- Opening Sentences*: The total number of opening sentences.
- Middle Sentences*: The total number of middle sentences.
- Closing Sentences*: The total number of closing sentences.

- g. *Discarded Sentences*: The total number of discarded sentences.
- h. *Original Text Readability*: The Flesch-Kinkaid reading ease score of the input text.
- i. *Summary Text Readability*: The Flesch-Kinkaid reading ease score of the summary text.
- j. *Part of Speech Tagging Used*: A boolean value whether the user had used enabled part-of-speech tagging method for term extraction.
- k. *Word Stemming Enabled*: Whether word stemming was enabled by the user for sentence linking.
- l. *Collocations Used*: Whether collocation analysis was enabled for intelligent term extraction.
- m. *Reference Corpus*: The reference corpus that was selected by the user as corpus reference.

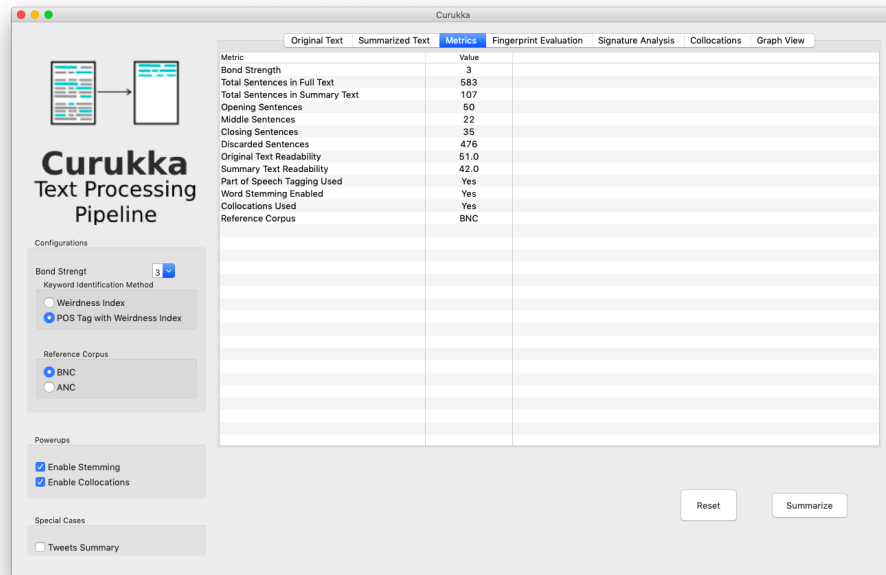


Figure 4.5: Screenshot of the Metrics Screen for the Curukka System

## Fingerprint Analysis View

The fingerprint analysis view renders the output of the cumulative relative frequency analysis (see section 3.4.3). The view is called fingerprint analysis view since the cumulative relative frequency generates a unique fingerprint word vectors based on their cumulative relative frequency. The view displays the results in a split-screen manner with the fingerprint of the input text in the left-hand side and the fingerprint of the summary text in the right-hand side. As explained in section 3.4.3, the fingerprint for the summary is generated based on the word vector of the top 100 open-class words in the input text. The output is rendered 10 words per row and in each row, the cumulative relative frequency up to that row is displayed to enable the user to examine the results. A screenshot of the fingerprint analysis view is shown in figure 4.6.

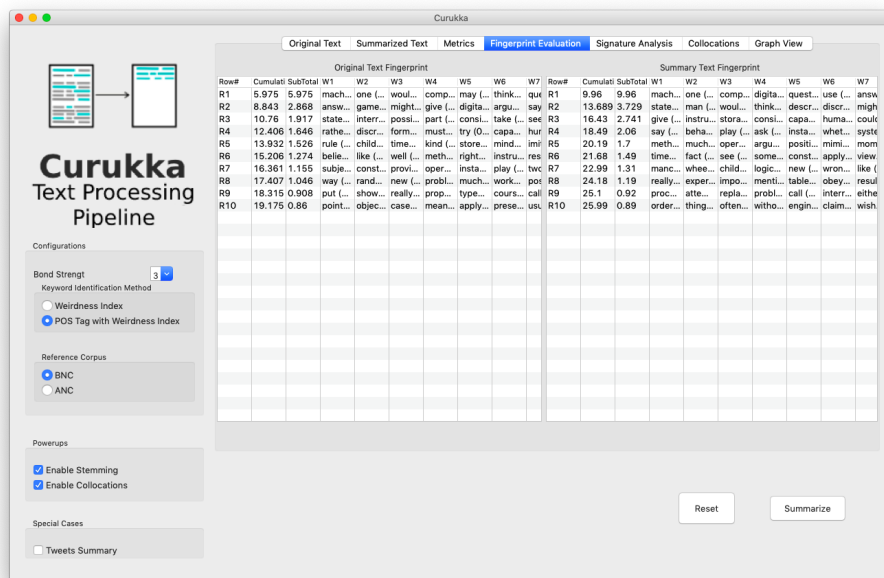


Figure 4.6: Screenshot of the Finger Print Analysis Screen for the Curukka System

## Signature Analysis View

The signature evaluation screen renders the output of the relative frequency analysis including the correlation analysis using relative frequency (see section 3.4.2). The view also enables the user to view the terms that were selected by the algorithm for creation



of the links between the sentences using intelligent term extraction (see section 3.3.1). In addition, the total number of tokens in the original text and summary text is also displayed in this view. All columns of this screen are sortable based on the user click on the column header and can be exported using the export option into a Comma Separated Value (CSV) file. A screenshot of the signature analysis screen is shown in figure 4.7.

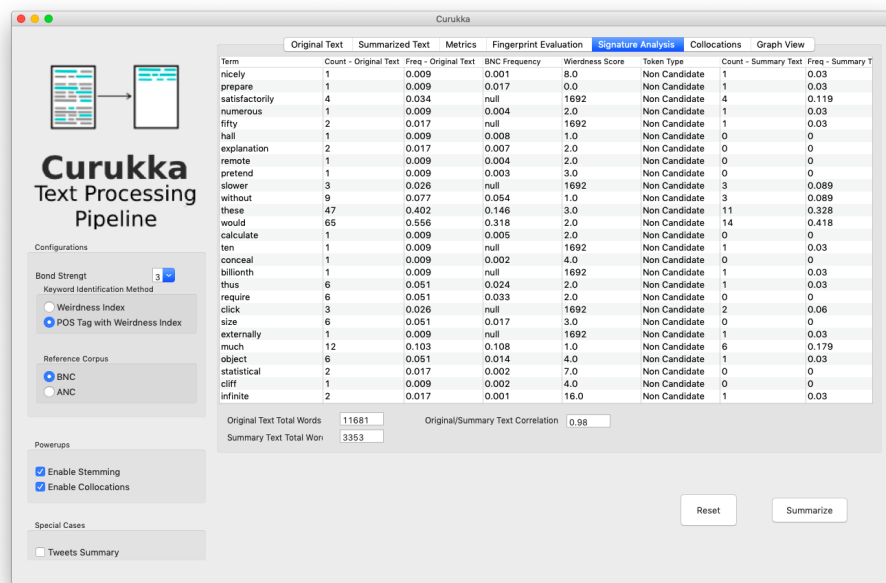


Figure 4.7: Screenshot of the Signature Analysis for the Curukka System

## Collocation Analysis View

The collocation analysis view renders the output of the collocation analysis (see section 10). Each of the columns in the view are sortable based on the user click on the column header. Also, the results rendered in this view can be exported to a CSV file using the export menu option. A screenshot of the collocation analysis view is shown in the figure 4.8.

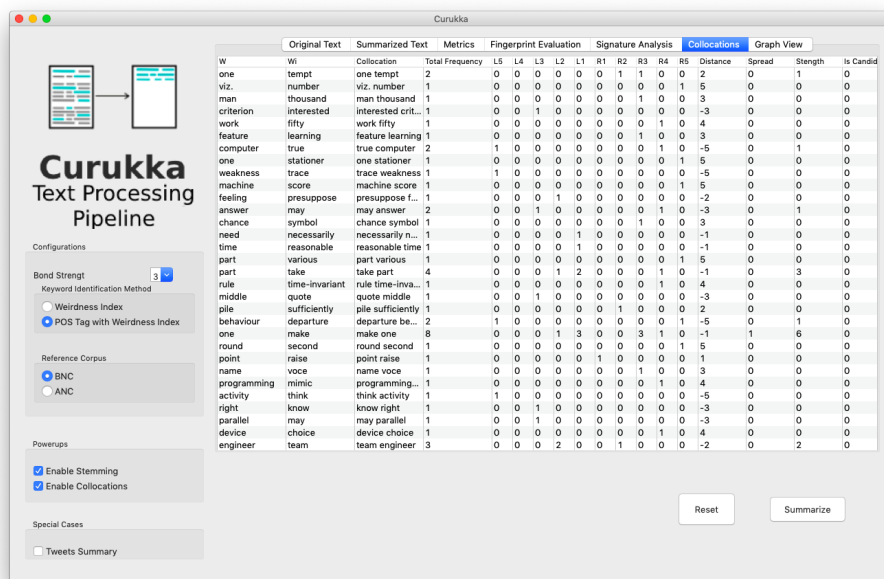


Figure 4.8: Screenshot of the Collocation Analysis Screen for the Curukka System

## Menu Options

The menu bar of the Curukka user interface application consists of two menu items: the file menu and the export menu.

A screenshot of the file menu is shown in the figure 4.9. The options of the file menu are as follows.

- Open*: The menu option to open a new text (.txt) file for summarization. It opens a file open dialog box.
- Save*: The menu option to save the summary to a text (.txt) file. It opens a save dialog box.
- Reset All*: The menu option to reset all options to factory settings.
- Exit*: The menu option to exit the Curukka application.

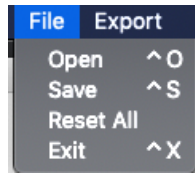


Figure 4.9: Screenshot of the File Menu option the Curukka System

The export menu provides the user with options to export the rendered outputs. A screenshot of the export menu is shown in the figure 4.10. The options available in the export menu are as follows.

- a. *Signature Analysis*: The export menu option to save the signature analysis into a CSV file. It opens a save file dialog box.
- b. *Collocations*: The export menu option to save the collocations into a CSV file. It opens up a dialog box.

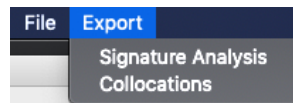


Figure 4.10: Screenshot of the Export Menu option in the Curukka System

Both the export menu items display a file export dialog box using which the user can choose the folder to where the file should be exported to. A screenshot of the file export dialog box is shown in figure 4.11.

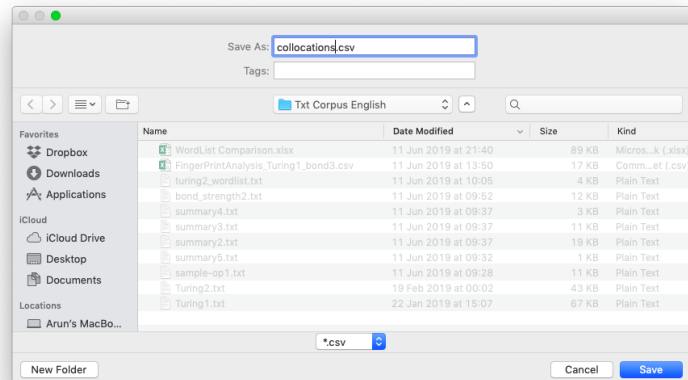


Figure 4.11: Screenshot of the Export CSV File dialogbox in the Curukka System

## 4.4 Case Studies and Discussion

This sections explains the case studies on the *Intelligent Text Summarization* algorithm and the discussion on the same.

### 4.4.1 Controlled Evaluation on Computer Science Text

A controlled evaluation was done using a dataset of computer science texts described in section 4.2. The figure 4.12 shows the comparison of the number of sentences in the input text and the summary text produced using the Intelligent Text Summarization algorithm plotted in a logarithmic scale. Similarly, figure 4.13 shows the comparison of the tokens in the input text and the summary text. The summarization was done with different combinations such as various bond strengths and configuration options. The best results were obtained when the bond strength was set as 3, POS tagging used with weirdness index, along with stemming and collocation analysis. Based on the observation from the controlled evaluation showcased in figure 4.12 and 4.13, it is observed that the average summary text produced by the algorithm is 28% of the input text on an average.

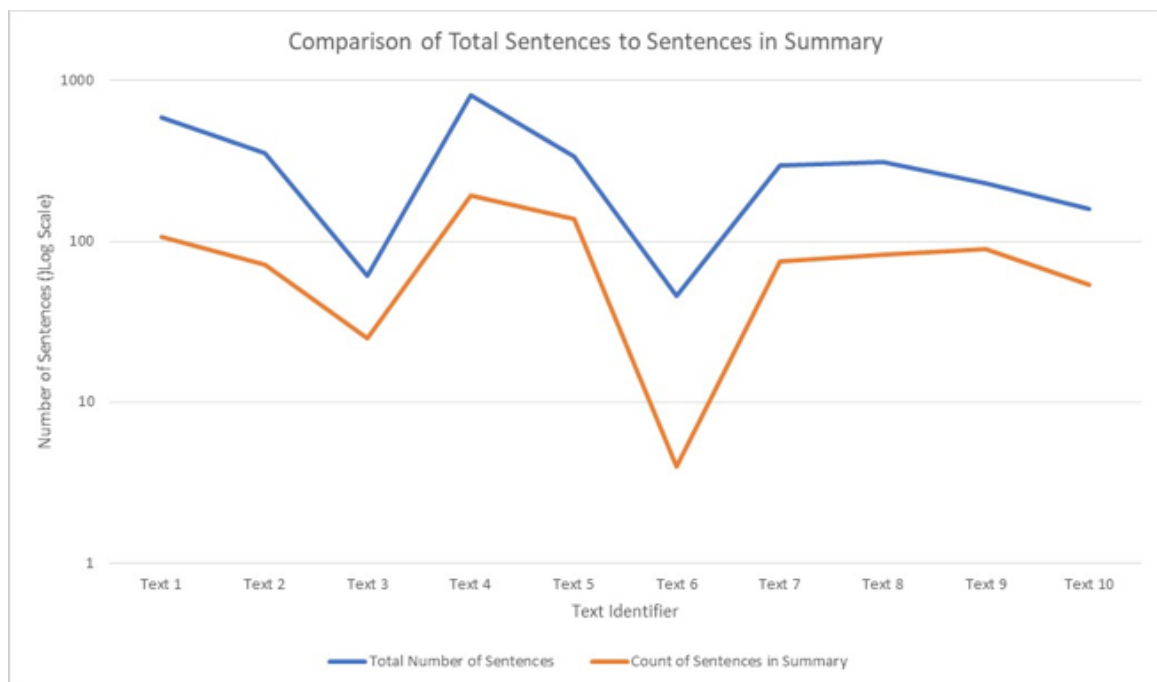


Figure 4.12: Comparison of total sentences of the original with the sentences obtained from proposed text summarization method

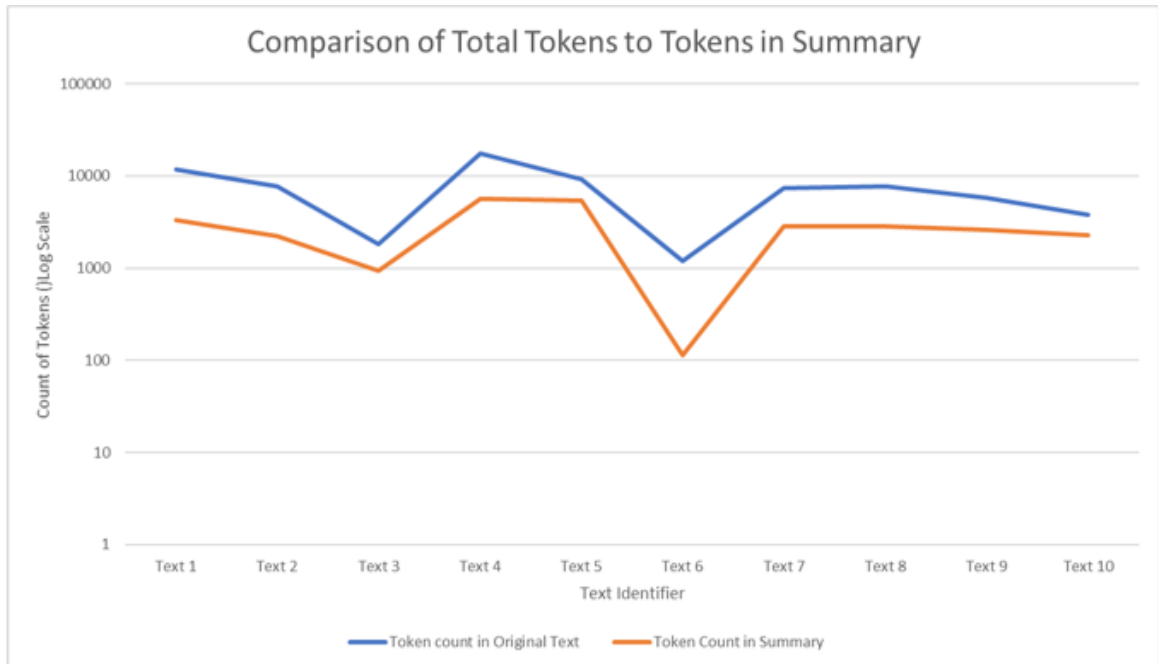


Figure 4.13: Comparison of total token count of the original with the sentences obtained from proposed text summarization method

**Readability Index Evaluation:** The readability index for the same set of computer science texts dataset was recorded for both the input text and summary text. The observations are visualized in figure 4.14. Based on the evaluation, the average readability index of the input text was 39, compared to the average readability index of 35.4 for the summary text. Based on the result it is established that the summaries produced by the intelligent text summarization algorithm do not decrease the readability. The small drop in the readability index average in summary text can be attributed to the fact that the readability index calculation depends on the ratio of words to sentences and syllables to words; since the summary has a major reduction in the vital calculation parameters, the average index score is slightly reduced.

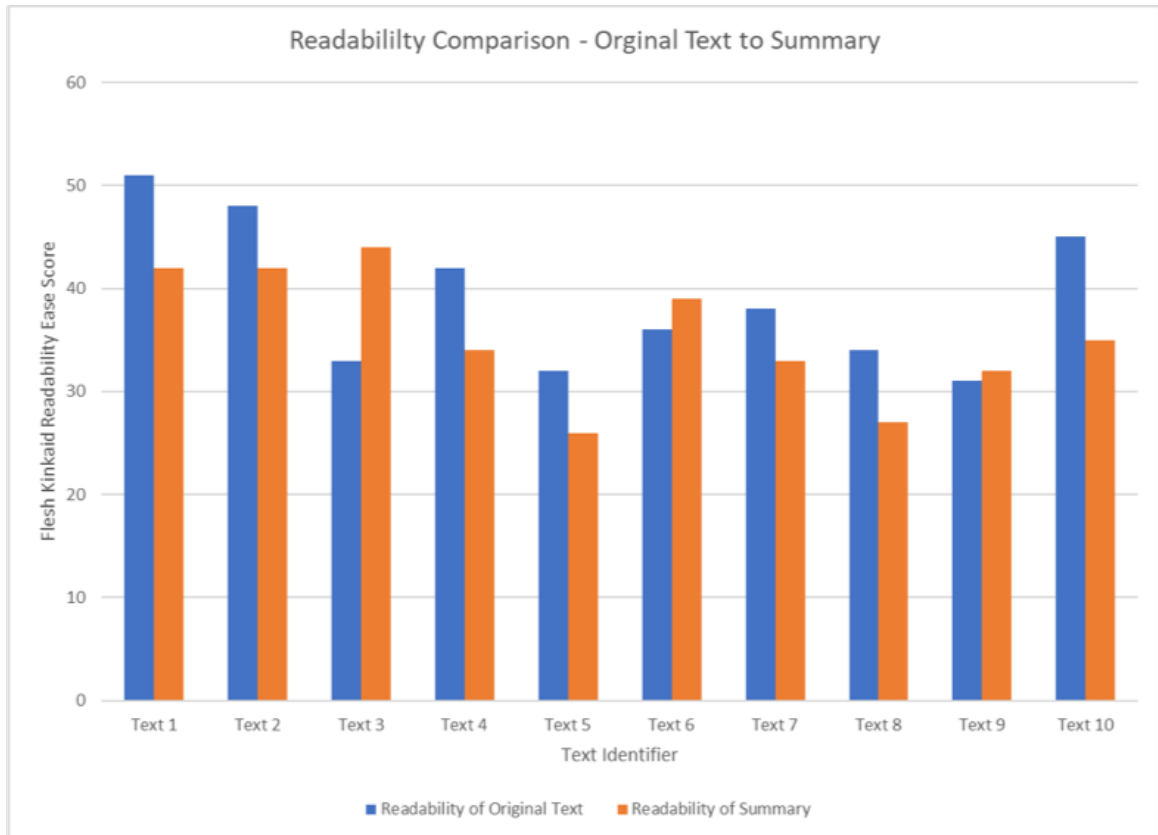


Figure 4.14: Comparison of readability index of original with the readability index of the summary text obtained from proposed text summarization method

***Cumulative Relative Frequency Evaluation:*** The final statistical evaluation of computer science dataset is based on the cumulative relative frequency. The cumulative relative frequency of the top 100 open-class words in both the input text and summary text is systematically recorded and visualized in the figure 4.15. Based on the observations, the average cumulative relative frequency of the top 100 open-class words in the input text was 21.84% and that of the summary text was 21.43%. The results stand a statistical proof that the information contained in the original text is retained in the summary text.

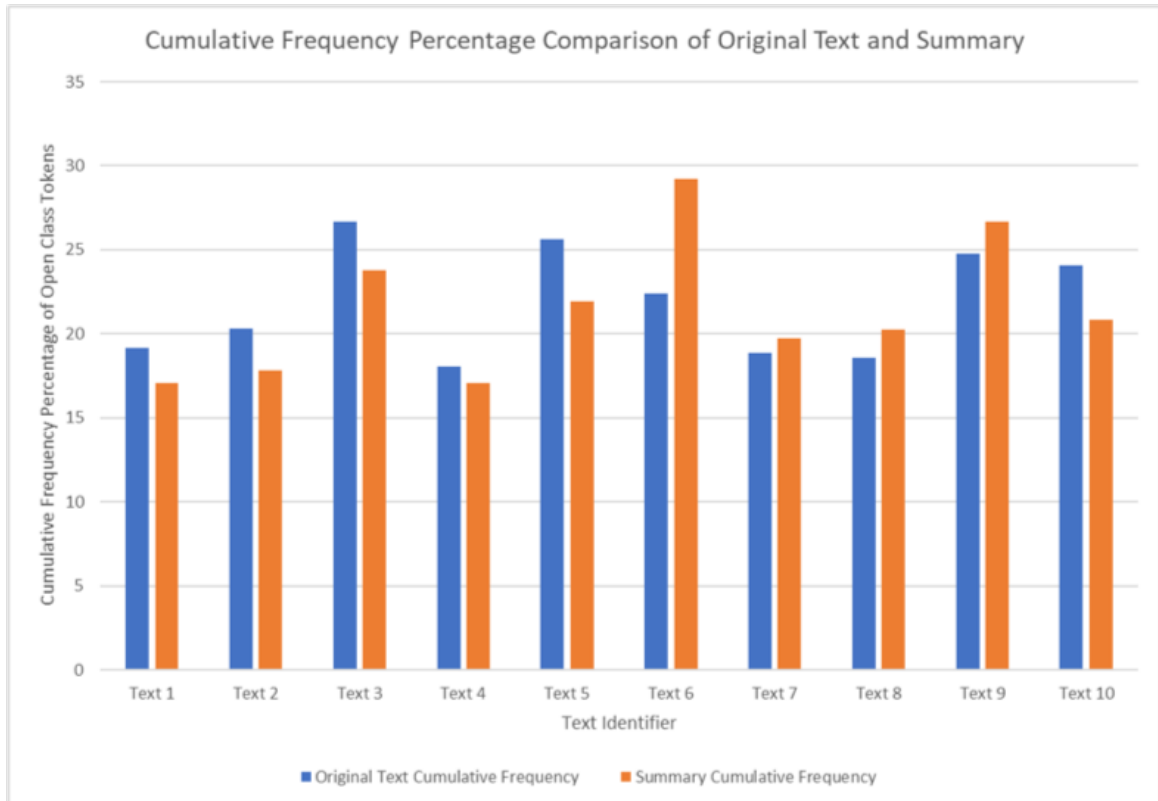


Figure 4.15: Comparison of cumulative relative frequency of original with the cumulative relative frequency of the summary text obtained from proposed text summarization method

#### 4.4.2 Controlled Evaluation on the Bio-medicine dataset

The automated evaluations similar to the ones conducted for the computer science dataset (see section 4.4.1) was performed on the bio-medicine dataset as well and the results were similar. However, the summaries produced by the intelligent text summarization algorithm in the bio-medicine dataset containing texts related to gastroenterology was also subjectively evaluated using a subject manner in bio-medicine. The summaries were evaluated on a score of 10. Aspects such as the quality of the summary produced, the ease of readability and understanding were considered as the grading criteria among others. The results are visualized in the figure 4.16. Based on the results recorded, the average score for the summaries generated as graded by the subject matter expert was 7.36 out of 10. This evaluation is vital to prove the



practicability of the proposed algorithm since the summaries are verified and graded by a human.

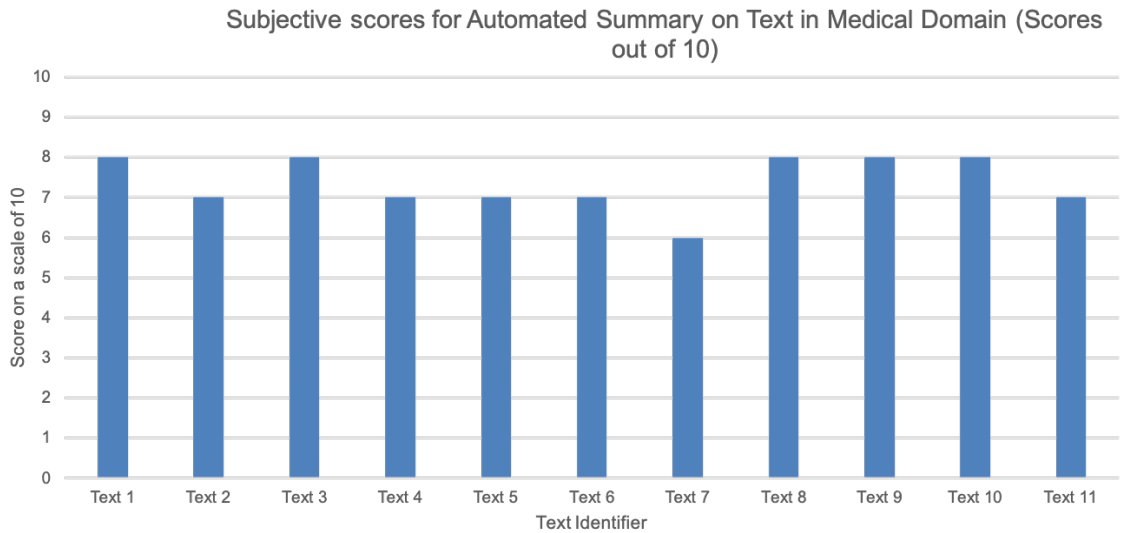


Figure 4.16: Subjective evaluation of the summaries produced for the bio-medicine dataset by the intelligent text summarization algorithm by a subject matter expert on a scale of 10

### 4.4.3 Experimental evaluation of summarization of large block of tweets

To test the practicability of the intelligent text summarization algorithm on other domains such as social media analytics, a tweet block containing fifty thousand tweets were summarized using the proposed algorithm. The tweet summarization configuration in the Curukka user interface was enabled. The input text consisted of the tweets recorded during the Russian River Flooding that happened in the Sonoma County, California, USA during March 2019. The summarization algorithm was able to autonomously extract key terms such as *flooding, river, disaster, Russian river, etc.*, and also create a summary in which the most important tweets were selected and included in the summary. The summary consisted of a total of sixteen thousand sentences. However, in the case of tweet summarization, a bond strength of 6 produced the best results. The results establish that the intelligent text summarization algorithm can be

extended for use in social media analytics with application areas such as emergency management.

## 4.5 System Performance

The performance of the system under various scenarios and load is important to gauge the practicability of the implementation in terms of scalability and reliability. Therefore the system was put under various modes of stress testing in order to understand the performance of the system.

Based on the experimentation done on a hardware with eight CPUs, 16 GB of RAM and 2.3 GHz processor with 4 GB of RAM allocated for the server and 8 GB of RAM allocated to the user interface, the system was able to gracefully handle text inputs with upto 4 million tokens. When the load was increased to 5 million tokens the user interface was able to send request to the server and receive response, however the user interface was not able to render the output. Hence, based on the controlled experimentation, it is noted that the system deployed on a hardware with the aforementioned specification can process and input text with upto 4 million tokens in it.

## 4.6 Security and Privacy Concerns

This work focuses on designing an intelligent text summarizer for specific language text and building an application that will be used to demonstrate the work. Free and open-source (FOSS) software is used to build the core components of the application to implement the designed algorithms.

In the case of the intelligent text summaries and with the use of FOSS software, there are associated security and privacy concerns.

### 4.6.1 Security Concerns

The security concerns related to the work can be broadly classified into three sections. Concerns related to the core algorithm implementation, with the use of the third-party software and libraries, and concerns related to the application being generally available.

## Concerns in Core Algorithm Implementation

The core algorithm that will stand as the bedrock of this research deals with the lexical cohesion analysis. In terms of the lexical cohesion using computation linguistics, the primary part is being able to read text using a programming language and apply cohesion analysis on it. In the scope of this work, an intelligent text summarizer is the objective, hence text from multiple languages have to be dealt with. With this in mind, it is necessary to consider that application takes into account the security issues related to the text input which can potentially compromise the working of the application providing false results or allows the user to manipulate the results. The attacks that the algorithm should be resistant to include the following areas.

***Encoding Related Security Issues:*** The application should be resistant towards attacks related to the text encoding. When a malicious user tries to break the algorithm or increase the time complexity of the application using combinations of encoding or by using control characters the application should be resistant towards it. An example of it may be that the malicious user provides a specific language text with control characters embedded between each letter of the text. Another example would be to mix the encoding of the text wrapping it using another encoding scheme using steganographic methods.

Without added security, the processing time of the algorithm will increase multi-fold and will break the system with such encoding related attacks. Hence, the application implementing the core algorithm should have a pre-processing module that will pre-process the input text and remove control characters, check for steganographic attacks and reject the request if any attacks are detected.

***Result Manipulation Security Issues:*** Another category of a security issue that is related to the core algorithm is to the manipulation of the results. If the results of the algorithm can be manipulated as per the user needs, then it poses a huge security risk. A way to do this in case of this work will be to use techniques such as keyword or key-phrase repetition. For example, repeating one paragraph of the text as-is or with slight changes, multiple times might make the algorithm biased towards the overloaded text thereby enabling the malicious user to manipulate the output.

This attack can be mitigated by algorithm side by making use of weighted averages and the majority of the mitigation can be done on the application implementation

where a pre-processing module can be used to detect repeated paragraphs. However, the problem becomes difficult when the paragraphs are not-repeated as-is but a random sentence rotation is used to jumble the repeated paragraphs.

### **Concerns with use of Free and Open Source Libraries**

According to Open Web Application Security Project (OWASP) Top Ten 2017, using components with known vulnerabilities is one of the top 10 most critical web security risks. However, it is inevitable to use components in application building with vulnerabilities since all or most of the applications and components available for use. It is no different with FOSS applications and components [56].

With respect to this work, it is essential to be aware of the vulnerabilities present in all the components that are core to the algorithm like the NLP components and the text processing libraries and to create a detailed risk assessment with the vulnerabilities, the probability of it occurring and the severity. Vulnerabilities related to the components used can be retrieved from an open vulnerability database such as Common Vulnerabilities and Exposure Details. Based on the descending order of severity of the vulnerability the mitigation plan should be created until there are no critical or high severity risks. Care should also be taken that the components used are configured in the right way with minimal attack surface. For example, for use of the Stanford NLP server, a server deployment mode with local binding should be used and the web server port should be blocked in the firewall to avoid external traffic ingress and only the local host should be whitelisted for the use of the server. Hence, only the application implementing the algorithm and co-hosted with the Stanford NLP server will be able to access it. If a distributed approach is followed, necessary security implementation in terms of firewall rules and whitelisting should be done to reduce the attack surface.

### **Concerns in making the Application Generally Available**

When the application is made generally available (GA) as a library or as a web application of consumption by the general public, additional security concerns arise.

***Packaging as a Library:*** In terms of packaging the algorithm as a library, periodic and automated security updates of the dependent components along with end-to-end integration testing should be part of the build, package and deploy pipeline.

This will ensure that the library is up-to-date in terms of security. In cases where the automated deployment fails, manual intervention will be required. Appropriate policies for security updates along with building and deploy instructions should be well documented as a part of the documentation and updated periodically in case the library is made available open-source.

***Deployment as a Web Application:*** In the case of deploying the application as a web application, all the security measures related to web security should be taken care of. Protection against well-known and commonly encountered issues such as injection, broken authentication, sensitive data exposure, security misconfiguration, cross-site scripting (XSS), insecure deserialization, insufficient logging, and monitoring should be prioritized and addressed.

Also, care should be taken to ensure that the application is protected from attacks such as Denial of Service (DOS), intrusion, etc., by deploying and configuring the appropriate defensive measures.

## 4.6.2 Privacy Concerns

The implementation of the algorithm as a library should be done with the privacy of the users in mind. Some of the measures that are to be taken to ensure privacy can be in terms of the core implementation of the algorithm as a library and making the application available as a web application for general use.

In terms of implementing the algorithm as a library, it should be ensured that no user data is sent over to any service or server without explicit consent from the end-user and the feature if exists (say for analytics of usage) should be turned off by default. Besides, all user-related information should be protected using secure compartmentalized storage with the use of encryption techniques for sensitive information. When the work is available as a web application for the use by the public, the data protection and privacy rules prevalent to the locality of deployment should be strictly implemented and followed.

## 4.7 Summary

This chapter provided a detailed view on the dataset used for evaluation, the architecture of the system implementation (see section 4.3.1) including the technical stack (see section 4.3.2) used, the user interface design (see section 4.3.3) and the results of the case studies (see section 4.4) carried out. This chapter establishes that the *Intelligent Text Summarization* algorithm proposed in this work is practicable and can be implemented as a working software to perform text summarization. The system performance (see section 4.5) in terms of scalability and reliability were also discussed in this chapter. The section on security and privacy (see section 4.6) explained in this chapter provides a careful examination of the concerns in the area of security and privacy when the application has to be deployed for general public use in production mode.

# Chapter 5

## Conclusion and Future Work

This chapter concludes the work summarizing the key take aways and lists out the limitations of the system which should be considered when this work is to be leveraged. This chapter also provides a section on suggestions for the future work that can be considered to extend this work.

### 5.1 Conclusion

An intelligent text summarization algorithm that can summarize large scientific texts of non-narrative nature is designed and implemented in this work. The intelligent text summarization algorithm designed in this work leverages the linguistic concept of lexical cohesion to identify the most important sentences in the scientific text and categories them into the topic opening, middle and closing sentences to provide a meaningful and organized summary of the input text without loss of information contained in the input scientific text. This work contributes to the field of Natural Language Processing in terms that the algorithms are capable of understanding the theme in scientific text and has a sub-module that will automatically understand and identify the important open-class terms in the scientific text without any manual input from the user and use it for linking the sentences in the input text as a directed graph hence can be attributed to the contribution in the field of artificial intelligence. The only input that is required from the user is the scientific text itself and the algorithm produces the summary of the text. This work also proposes and implements automated evaluation algorithms

for the determination of the goodness of the summary using statistical measures which substantially reduces the need for human-intensive evaluation.

A detailed design specification and implementation architecture are provided in this work where the proposed intelligent text summarization algorithm is implemented along with the evaluation algorithms as a working software that can be readily used for summarization and evaluation of the summaries produced. The implementation is used to test the algorithm and evaluate its working in real-life. The implementation is also evaluated for scalability and reliability and the system performance is documented.

Using the implementation of the system, controlled evaluation is done on the proposed intelligent text summarization algorithm. The automated evaluation results and the subjective human evaluation results are documented. Based on the controlled evaluation the result observed is that bond strength of three works the best for scientific texts from various domains and the average summary size is 28% of the input text. Based on the controlled subjective evaluation by a human with expertise in the domain of the articles used for summarization, the average score for summarization on a scale of 10 is 7.36. This work also showcases experimentation where the large volumes of tweets (fifty thousand) are summarized in order to extract the most important tweets. The experimentation with tweets opens up the scope that the intelligent text summarization algorithm proposed in this work can be adapted for application areas such as emergency management using social media analytics.

## 5.2 Future Work

This section list out the limitations of the intelligent text summarization algorithm that were identified during the implementation and the controlled evaluation. Future work based on this algorithm needs to consider these limitations to make an informed decision on the usage and extension of the intelligent text summarization algorithm contributed by this work.

1. The algorithm has been implemented and evaluated for large scientific texts in the English language. Though the algorithm is not specific to the language and can work with other Indo-germanic languages with the syntactic structure of English, it has not been tested or evaluated due to the shortage of time and resources.



Further, for the algorithm to work with other languages, a reference corpus for the language should be available or a new one should be built. Trying the algorithm with other languages such as Tamil (an ancient Dravidian language) is a potential future work.

2. Some of the routines in the algorithm have a time complexity of  $n^2$ . There is a scope to optimize these routines so that the time complexity of the algorithm can be reduced enabling better scalability. This can be a future work where the algorithm can be optimized and extensive load testing done to evaluate its performance.
3. At the time of writing this thesis, it was not able to compare the advantage of autonomous term extraction over systems without it for text summarization. Based on the study, the autonomous text summarization systems could be build. This work is in progress as a journal article [57].
4. This work showcases experimentation where the algorithm is used to summarize large blocks of tweets with application scope in the field of social media analytics. However, extensive studies need to be conducted to test the practicability and required changes to the algorithm and the implementation of the algorithm for the use in social media analytics.

Apart from the aforementioned limitations and scope for future work, researchers planning to make use of the algorithm and/or its implementation should consider the security and privacy concerns when the algorithm is used for public-facing or machine critical applications. The section on security and privacy (see section 4.6) provides a detailed discussion on the concerns with respect to the current state of the algorithm and implementation, however, it needs to be reviewed and amended when the work is extended on leveraged.

# Bibliography

- [1] K. D. Bromley and L. McKeveny, “Précis writing: Suggestions for instruction in summarizing,” *Journal of Reading*, vol. 29, no. 5, pp. 392–395, 1986.
- [2] M. Benhrahim and K. Ahmad, “Text summarisation: The role of lexical cohesion analysis,” *The New Review of Document & Text Management*, pp. 321–335, 1995.
- [3] A. Saseendran and K. Ahmad, “Intelligent text summarization: Leveraging cohesion in text,” *Work in Progress - Yet to be published*.
- [4] R. Raso, D. Werth, and P. Loos, “Enriching augmented reality with text data mining: An automated content management system to develop hybrid media applications,” in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.
- [5] D. Bamman and N. A. Smith, “New alignment methods for discriminative book summarization,” *arXiv preprint arXiv:1305.1319*, 2013.
- [6] A. Abdi, N. Idris, R. M. Alguliev, and R. M. Aliguliyev, “Automatic summarization assessment through a combination of semantic and syntactic information for intelligent educational systems,” *Information Processing & Management*, vol. 51, no. 4, pp. 340–358, 2015.
- [7] M. Yousefi-Azar and L. Hamey, “Text summarization using unsupervised deep learning,” *Expert Systems with Applications*, vol. 68, pp. 93–105, 2017.
- [8] S. Wang, X. Zhao, B. Li, B. Ge, and D. Tang, “Integrating extractive and abstractive models for long text summarization,” in *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 305–312, IEEE, 2017.

- [9] K. Nandhini and S. R. Balasundaram, “Improving readability of dyslexic learners through document summarization,” in *2011 IEEE International Conference on Technology for Education*, pp. 246–249, IEEE, 2011.
- [10] A. Farzindar and G. Lapalme, “Legal text summarization by exploration of the thematic structure and argumentative roles,” in *Text Summarization Branches Out*, pp. 27–34, 2004.
- [11] S. Afantenos, V. Karkaletsis, and P. Stamatopoulos, “Summarization from medical documents: a survey,” *Artificial intelligence in medicine*, vol. 33, no. 2, pp. 157–177, 2005.
- [12] G. Murray, S. Renals, and J. Carletta, “Extractive summarization of meeting recordings,” 2005.
- [13] Y.-H. Tseng, C.-J. Lin, and Y.-I. Lin, “Text mining techniques for patent analysis,” *Information Processing & Management*, vol. 43, no. 5, pp. 1216–1247, 2007.
- [14] D. D. A. Bui, G. Del Fiol, J. F. Hurdle, and S. Jonnalagadda, “Extractive text summarization system to aid data extraction from full text in systematic review development,” *Journal of biomedical informatics*, vol. 64, pp. 265–272, 2016.
- [15] D. Thorleuchter and D. Van den Poel, “Using text summarizing to support planning of research and development,” in *New Perspectives in Information Systems and Technologies, Volume 1*, pp. 23–29, Springer, 2014.
- [16] C.-W. Wu and C.-L. Liu, “Ontology-based text summarization for business news articles,” *Computers and their applications*, vol. 2003, pp. 389–392, 2003.
- [17] B. Freeling, Z. A. Doubleday, and S. D. Connell, “Opinion: How can we boost the impact of publications? try better writing,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 2, pp. 341–343, 2019.
- [18] P. Plavén-Sigray, G. J. Matheson, B. C. Schiffler, and W. H. Thompson, “The readability of scientific texts is decreasing over time,” *Elife*, vol. 6, p. e27725, 2017.

- [19] D. Crystal, *A dictionary of linguistics and phonetics*. Cambridge Blackwell Publication, 2003.
- [20] M. A. K. Halliday and R. Hasan, *Cohesion in english*. Routledge, 2014.
- [21] M. Hoey, *Patterns of lexis in text*, vol. 299. Oxford University Press Oxford, 1991.
- [22] C. D. Paice and P. A. Jones, “The identification of important concepts in highly structured technical papers,” in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 69–78, ACM, 1993.
- [23] G. Erkan and D. R. Radev, “Lexrank: Graph-based lexical centrality as salience in text summarization,” *Journal of artificial intelligence research*, vol. 22, pp. 457–479, 2004.
- [24] Y. Lin, “A study of cohesion in simultaneous interpreting,” in *2019 International Conference on Contemporary Education and Society Development (ICCESD 2019)*, Atlantis Press, 2019.
- [25] L. Antiquiera, O. N. Oliveira Jr, L. da Fontoura Costa, and M. d. G. V. Nunes, “A complex network approach to text summarization,” *Information Sciences*, vol. 179, no. 5, pp. 584–599, 2009.
- [26] I. da Cunha, E. SanJuan, J.-M. Torres-Moreno, and I. Castellón, “Extending automatic discourse segmentation for texts in spanish to catalan,” *arXiv preprint arXiv:1703.04718*, 2017.
- [27] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell, “Summarizing text documents: sentence selection and evaluation metrics,” in *SIGIR*, vol. 99, p. 99, 1999.
- [28] C. Blake and T. Rindflesch, “Leveraging syntax to better capture the semantics of elliptical coordinated compound noun phrases,” *Journal of biomedical informatics*, vol. 72, pp. 120–131, 2017.
- [29] V. M. Vujević, “Ellipsis and substitution as cohesive devices,” 2012.

- [30] R. Mihalcea and P. Tarau, “Graph-based ranking algorithms for text processing,” Oct. 5 2010. US Patent 7,809,548.
- [31] J. Diesner and K. M. Carley, “Revealing social structure from texts: meta-matrix text analysis as a novel method for network text analysis,” in *Causal mapping for research in information technology*, pp. 81–108, IGI Global, 2005.
- [32] D. Wang, T. Li, S. Zhu, and C. Ding, “Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 307–314, ACM, 2008.
- [33] F. B. Goularte, S. M. Nassar, R. Fileto, and H. Saggion, “A text summarization method based on fuzzy rules and applicable to automated assessment,” *Expert Systems with Applications*, vol. 115, pp. 264–275, 2019.
- [34] J. Cheng and M. Lapata, “Neural summarization by extracting sentences and words,” *arXiv preprint arXiv:1603.07252*, 2016.
- [35] R. Kumar, H. S. Pannu, and A. K. Malhi, “Aspect-based sentiment analysis using deep networks and stochastic optimization,” *Neural Computing and Applications*, pp. 1–15, 2019.
- [36] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, “Universal dependency parsing from scratch,” in *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, (Brussels, Belgium), pp. 160–170, Association for Computational Linguistics, October 2018.
- [37] A. Khurshid, L. Gillman, and L. Tostevin, “Weirdness indexing for logical document extrapolation and retrieval,” in *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, 2000.
- [38] R. De Beaugrande and W. U. Dressler, *Introduction to text linguistics*. Routledge, 1981.
- [39] U. of Rhode Island, “Introduction to Computers.” <https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading01.htm>, 2019. [Online; accessed 10-August-2019].

- [40] ISO/IEC, “Final Text of DIS 8859-1, 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No.1.” <http://www.open-std.org/JTC1/SC2/WG3/docs/n411.pdf>, 1998. [Online; accessed 10-August-2019].
- [41] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.
- [42] M. Rogers and K. Ahmad, “Corpus linguistics and terminology extraction,” *Sue-Ellen Wright and Gerhard Budin. Handbook of Terminology Management, Amsterdam Philadelphia*, vol. 2, pp. 725–760, 2001.
- [43] N. Ide and C. Macleod, “The american national corpus: A standardized resource of american english,” in *Proceedings of corpus linguistics*, vol. 3, pp. 1–7, Lancaster University Centre for Computer Corpus Research on Language , 2001.
- [44] G. Leech, P. Rayson, *et al.*, *Word frequencies in written and spoken English: Based on the British National Corpus*. Routledge, 2014.
- [45] A. Kilgarriff, “BNC database and word frequency lists.” <http://www.kilgarriff.co.uk/bnc-readme.html>, 1996. [Online; accessed 10-August-2019].
- [46] F. Smadja, “Retrieving collocations from text: Xtract,” *Computational linguistics*, vol. 19, no. 1, pp. 143–177, 1993.
- [47] M. F. Porter, R. Boulton, and A. Macfarlane, “The english (porter2) stemming algorithm,” *Retrieved*, vol. 18, p. 2011, 2002.
- [48] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, “Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel,” 1975.
- [49] N. Cheshire and H. Thomä, “Metaphor, neologism and ‘open texture’: implications for translating freud’s scientific thought,” *International Review of Psycho-Analysis*, vol. 18, pp. 429–455, 1991.
- [50] A. TURING, “I.–computing machinery and intelligence,” *Mind*, vol. 59, no. 236, pp. 433–433, 1950.

- [51] A. M. Turing, “On computable numbers, with an application to the entscheidungsproblem,” *Proceedings of the London mathematical society*, vol. 2, no. 1, pp. 230–265, 1937.
- [52] A. TURING, “The chemical basis of morphogenesis,” *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 237, no. 641, pp. 37–72, 1952.
- [53] M. Minsky, “Steps toward artificial intelligence,” *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.
- [54] E. Mollick, “Establishing moore’s law,” *IEEE Annals of the History of Computing*, vol. 28, no. 3, pp. 62–75, 2006.
- [55] O. M. Bonastre and A. Veà, “Origins of the domain name system,” *IEEE Annals of the History of Computing*, vol. 41, no. 2, pp. 48–60, 2019.
- [56] S. R. Vadalasetty, “Security concerns in using open source software for enterprise requirements,” *SANS Institute*, 2003.
- [57] A. Saseendran and K. Ahmad, “Towards autonomous text summarization,” *Work in Progress - Yet to be published*.

# Appendix

## I. Dataset Used for Evaluation

### Computer Science Dataset

The titles of the articles in the computer science dataset is given below.

- Computing Machinery and Intelligence  
(<http://cogprints.org/499/1/turing.html>, online, accessed 10-August-2019)
- Lecture to the London Mathematical Society  
(<https://www.vordenker.de/downloads/turing-vorlesung.pdf>, online, accessed 10-August-2019)
- Turing Invents the Universal Turing Machine  
(<https://dl.acm.org/citation.cfm?id=1391235>, online, accessed 10-August-2019)
- Why the Arpanet was built  
(<https://ieeexplore.ieee.org/abstract/document/5432117>, online, accessed 10-August-2019)
- The Materiality of the Internet  
(<https://ieeexplore.ieee.org/document/1677468>, online, accessed 10-August-2019)
- The ENIACs 1949 Determination of PI  
(<https://ieeexplore.ieee.org/abstract/document/5999630>, online, accessed 10-August-2019)
- Origins of the DNS  
(<https://ieeexplore.ieee.org/document/8700196>, online, accessed 10-August-2019)



- IBM Relational Database Systems The Early Years  
(<https://ieeexplore.ieee.org/document/6297962>, online, accessed 10-August-2019)
- Establishing Moores Law  
(<https://ieeexplore.ieee.org/document/1677462>, online, accessed 10-August-2019)

## **Bio-medicine Dataset**

The titles of the texts used from the bio-medicine domain is given below

- Diagnosis of gastrointestinal bleeding: A practical guide for clinicians
- Epidemiology of acute upper gastrointestinal bleeding
- The Overall Approach to the Management of Upper Gastrointestinal Bleeding
- Systematic reviews of the clinical effectiveness and cost-effectiveness of proton pump inhibitors in acute upper gastrointestinal bleeding
- Restrictive vs liberal transfusion for upper gastrointestinal bleeding: A meta-analysis of randomized controlled trials
- Red cell transfusion for the management of upper gastrointestinal haemorrhage
- Gastrointestinal Bleeding
- Impact of More Restrictive Blood Transfusion Strategies on Clinical Outcomes: A Meta-analysis and Systematic Review
- Medical Management of Variceal Hemorrhage
- Does This Patient Have a Severe Upper Gastrointestinal Bleed?
- Effect of pharmacological therapies for stroke prevention on major gastrointestinal bleeding in patients with atrial fibrillation
- Nasogastric Aspiration and Lavage in Emergency Department Patients with Hematochezia or Melena Without Hematemesis

- Usefulness of CT angiography in diagnosing acute gastrointestinal bleeding: A meta-analysis
- International Consensus Recommendations on the Management of Patients With Nonvariceal Upper Gastrointestinal Bleeding
- Endoscopic band ligation versus pharmacological therapy for variceal bleeding in cirrhosis: A meta-analysis

## II. Intelligent Text Summarization Server Setup

The server can be setup on any linux server with a minimum of 8 GB of RAM, 8 CPUs and 20GB of Harddisk with JRE 1.8 and Python 3.7 installed. Ubuntu 18.04 is suggested. The below steps have to be followed.

1. Download and start the Stanford NLP server from <https://stanfordnlp.github.io/CoreNLP/download.html>
2. Unpack and start the server using the command  

```
'java -mx4g -cp "*" edu.stanford.nlp.pipeline.StanfordCoreNLPServer -annotators "tokenize,ssplit,pos,lemma,parse,sentiment" -port 9000 -timeout 30000'
```
3. Download the Curukka Server by cloning the git repository <https://github.com/ats0stv/TextSummarization.git>
4. Navigate to the folder "pythonCode" in the repository.
5. Execute the command "pip install -r requirements.txt"
6. Start the server using the command  

```
"FLASK_APP=TextSummarizer.py FLASK_DEBUG=1 python -m flask run"
```

The Curukka server is all setup and ready to serve requests.

### III. Curukka User Interface Setup

A computer with Microsoft Windows 10+ or Apple OSX 10.14+ operating system with atleast 4 CPU, 4 GB of RAM and 10 GB of harddisk space is the minimum requirement for the Curkka UI.

To run the Curukka user interface, follow the below steps.

1. Clone the git repository  
<https://github.com/ats0stv/TextSummarization.git>
2. Navigate to the GUI folder.
3. Double click on the Curukka.jar

Following the about steps will launch the Curukka User Interface application.