**CHAPTER 1**

**INTRODUCTION**

Wikipedia, the free online encyclopedia, ranked the fifth most popular website is widely used for learning. However, without basic knowledge of the subject, understanding a Wikipedia page is a very tedious task. We experienced the same after opening the Quantum Mechanics article – the first paragraph had abstruse terms like “quantization” and “probability amplitude”. We then went to the quantization page, where once again, we found many esoteric terms. After about thirty minutes, we were totally lost! Also, we were unable to find any other existing solution to make learning from Wikipedia less taxing.

To tackle this problem, we are developing a system which uses keyword extraction, to extract potentially difficult-to-understand terms from the article, text summarization to summarize these concepts and finally present this as prerequisites to the reader. After reading the prerequisites, the reader can understand the main article easily. This not only promotes fast reading but also smart reading. We face challenges in identifying the important keywords and presenting the pre-requisites in a grammatically acceptable form. The solution proposed by us is unique and can be implemented as a Wikipedia add-on. It can also be extended to other literature like technical papers and textbooks in the future.

Text summarization and keyword extraction form the foundation of the proposed solution. Automatic or text summarization is the process of shortening a text document with software, to create a summary with the major points of the original document. There are two general approaches: extraction and abstraction. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might express. Keyword extraction is tasked with the automatic identification of terms that best describe the subject of a document. Methods for automatic keyword extraction can be supervised, semi-supervised, or unsupervised. Having evaluated all options, we have used extractive text summarization and Rapid Automatic Keyword Extraction (RAKE) for keyword extraction. We have evaluated the performance of our system on Wikipedia page of famous personalities.

The report is organized in following manner - Chapter 2 deals with literature survey where we have described the papers that form the basis for our proposed system. Chapter 3 deals with system analysis where the existing system has been compared with the proposed system. Chapter 4 lists the system requirements and the system design has been described in Chapter 5 and 6. The detailed implementation is listed in Chapter 7, followed by testing in Chapter 8.

**CHAPTER 2**

**LITERATURE SURVEY**

Research on keyword extraction dates to the 70’s when researchers first tried devising engines that automatically return keywords for the given document. The key to keyword extraction involves breaking the article down to candidate keywords using a “stoplist” – which comprises of words like *and*, *the* and *of.* Research on text summarization has boomed over the last two decades and numerous implementation have been proposed, broadly classified under extractive and abstractive categories. Our proposed system has been inspired by the works listed below:

1. **“Automatic Keyword extraction from Individual Documents” by Stuart Rose, David Engel, Nick Cramer**

**Description:**

Rapid Automatic Keyword Extraction (RAKE), which was introduced in this paper is an unsupervised, domain and language independent method for extracting keywords from documents. RAKE is based on the simple observation that keywords seldom contain stop words – such as and, of and the. RAKE uses a list of stop words to split the document text into candidate keywords. The candidates are then evaluated and scored, where in a higher score, points to a more important keyword with respect to the given input document.

1. **“Multi-document summarization using document set type classification” by Jun’ichi Fukumoto and Tomoya Sugimura  
   Description:**This paper proposes a summarization system which automatically classifies type of document set and summarizes a document set with its appropriate summarization mechanism. This system will classify a document set into three types: (a) One topic type, (b) multi-topic type, and (c) others. These types will be identified using information of high frequency nouns and Named Entity. In out multi-document summarization system, unnecessary parts are deleted after summarizing is generated. In type (a), unnecessary parts are similar part between summarized documents by

single document summarization. In type (b), unnecessary parts are not similar part in the document. In type (c), unnecessary parts are identified by scores used for single document summarization.

1. **“Automatic text structuring and summarization” by Gerard Salton, Amit Singhal, Mandar Mitra and Chris Buckley  
   Description:**This paper explains the working of an Information Retrieval system (IR) which is a system that allows to find documents relevant to a user request, from a base voluminous document. In this definition, there are three key concepts: documents, query and relevance. Document: A document may be a text, a piece of text, a web page, an image, a video, etc. Query: A query expresses the need for user information. The objective of IR is to find only relevant documents. It considers three possible approaches: a document that contains that string, approach based on indexing and the notion of relevance. The paper also defines accuracy and reminder of a document. Relationship of IR with other fields such as databases and question-answering systems is also addressed.
2. **“A Survey on abstractive text summarization” by N. Moratanch, S. Chitrakala  
   Description:**Different abstractive text summarization techniques have been surveyed in this paper. Both structure-based approaches and semantic based approaches have been considered and compared. This survey portrays that most of the abstractive summarization methods produces highly cohesive, coherent, less redundant information rich summary. It was concluded that challenges in terms of time and space complexity have not been addressed in any of the works.
3. **“Automatic Extraction of Document Keyphrases for Use in Digital Libraries: Evaluation and Applications” by Steve Jones and Gordon W. Palmer  
   Description:**This article describes an evaluation of the Kea automatic keyword extraction algorithm. Document keywords are conventionally used as succinct descriptors of document content, and are increasingly used in innovative ways, including document clustering, searching, and retrieval engines. It is however costly and time consuming to manually assign keyphrases to documents, encouraging the development of tools that automatically perform this task. Previous works have evaluated Kea's performance by calculating its ability to identify author keywords and keyphrases, but this technique has several limitations. The results presented in this article are based on evaluations by human evaluators of the quality and correctness of Kea keyphrases. The results indicate that, overall, Kea produces keyphrases that are rated positively by human evaluators. However, typical Kea settings relating to keyphrase length and domain specificity can degrade performance. It was found that for some settings, Kea's performance is better than that of comparable systems, and that Kea's ranking of extracted keyphrases is effective.
4. **“Keyword extraction from a single document using word co-occurrence statistical information” by Matsuo Y and Ishizuka M  
   Description:**A new keyword extraction algorithm that applies to a single document without using a corpus was presented in this paper. After frequent terms are extracted, a set of cooccurrence between each term and the frequent terms, i.e., occurrences in the same sentences, is generated. Co-occurrence distribution shows the rank of a term in the document as follows - if the probability distribution of co-occurrence between term ‘a’ and the frequent terms is predisposed to a subset of frequent terms, then term ‘a’ is likely to be a keyword. The algorithm shows similar performance to tfidf without using a corpus.
5. **“Stop-words in keyphrase extraction problem” by S. Popova, L. Kovriguina, D. Mouromtsev, I. Khodyrev  
   Description:**Keyword extraction problem is one of the most significant tasks in information retrieval. High-quality keyword extraction sufficiently influences the progress in the following subtasks of information retrieval: classification and clustering, data mining, knowledge extraction and representation, etc. The research environment has specified a layout for keyphrase extraction. However, some of the possible decisions remain uninvolved in the paradigm. In the paper the authors observe the scope of interdisciplinary methods applicable to automatic stop list feeding. The chosen method belongs to the class of experiential models. The research procedure based on this method allows to improve the quality of keyphrase extraction on the stage of candidate keyphrase building. Several ways to automatic feeding of the stop lists are proposed in the paper as well. One of them is based on provisions of lexical statistics and the results of its application to the discussed task point out the non-gaussian nature of text corpora. The second way based on usage of the Inspec train collection to the feeding of stop lists improves the quality considerably.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

There are no automated summarization systems which provide the functionality of annotations for Wikipedia articles. Hence, we compare the system with the other related systems which were discussed earlier in the Literature Survey.

**Limitations of Existing System:**

* **Drawbacks of Salton’s Method:** There are problems with redundant words that may appear more than once and may seem relevant but aren’t. For example, the educational institute may be mentioned more than once and is not vital, but the algorithm considers otherwise.
* **Drawback of Fukumoto’s Method:** In the evaluation, the mechanism of document set classification does not work well. This is because the current implementation has some system bugs in classification mechanism. Although the classification method does not work well, the results of short and long extractions were not too bad. It is necessary to evaluate the mechanism after fixing system bugs and improvement of our system.

**3.2 PROPOSED SYSTEM**

We aim to create a system, that provides the user, a summary of the important topics which they need to know, for better understanding of the Wikipedia article currently being read. Our implementation is divided into two steps:

* Identification of keywords
* Summarization of the keywords

Keywords of the page which are important to the understanding of the article are identified. These concepts are then summarized by using the content of the respective Wikipedia page. This information is presented as pre-requisites to the user. For testing, we have limited our test space to articles on famous personalities.

Advantages of our system:

* Help the user understand the document better by giving a short, easy to read summary of the pre-requisites.
* It reduces the time taken to identify and obtain information about the prerequisites.

**Problem Statement**

Given a Wikipedia page on famous personalities, identify the terms whose understanding is vital to understanding the page as a whole, and present a succinct summary explaining these terms.

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

**4.1 HARDWARE REQUIREMENTS**

* Laptop/Desktop with active Internet - The system should be enabled with active high-speed internet (10Mbps++).
* Minimum Hardware
  + Processor - Pentium i5 and higher versions.
  + RAM - 4 to 8 GB

**4.2 SOFTWARE REQUIREMENTS**

* Operating System - Since the tool is a command-line tool, compatible with Linux or MacOS.
* Language and Libraries -
  + Python3 - All the libraries imported run on Python3.
  + Beautiful Soup - Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with lxml to provide idiomatic ways of navigating, searching, and modifying the parse tree. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping. [8]
  + NLTK - The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning.NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities. [9]
  + NumPy - NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. NumPy is an open-source software and has many contributors. Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations. [10]
  + Urllib - The urllib module in Python 3 allows access websites via your program. Through urllib, one can access websites, download data, parse data, modify your headers, and do any GET and POST requests one might need to do. [11]

**CHAPTER 5**

**SYSTEM DESIGN**

**5.1 INTRODUCTION**

System design is a blueprint of the solution for a system. Design of the system can be defined as the process of applying different techniques and principles to defining a process in sufficient detail to permit its physical realization. System design is concerned with how the system functionalities must be provided by its different components. The design activity often results in -

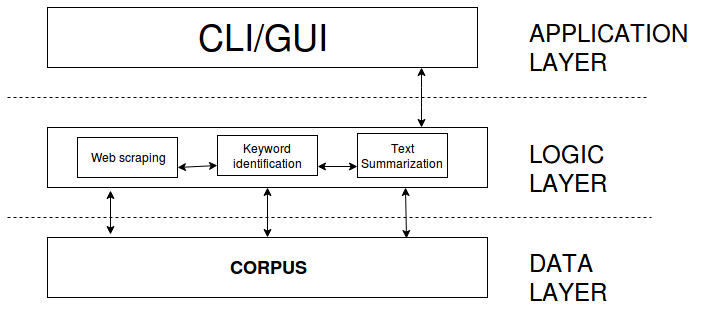
• Architecture

• High level design.

**5.2 ARCHITECTURE**

The proposed system contains a three-tier architecture, namely

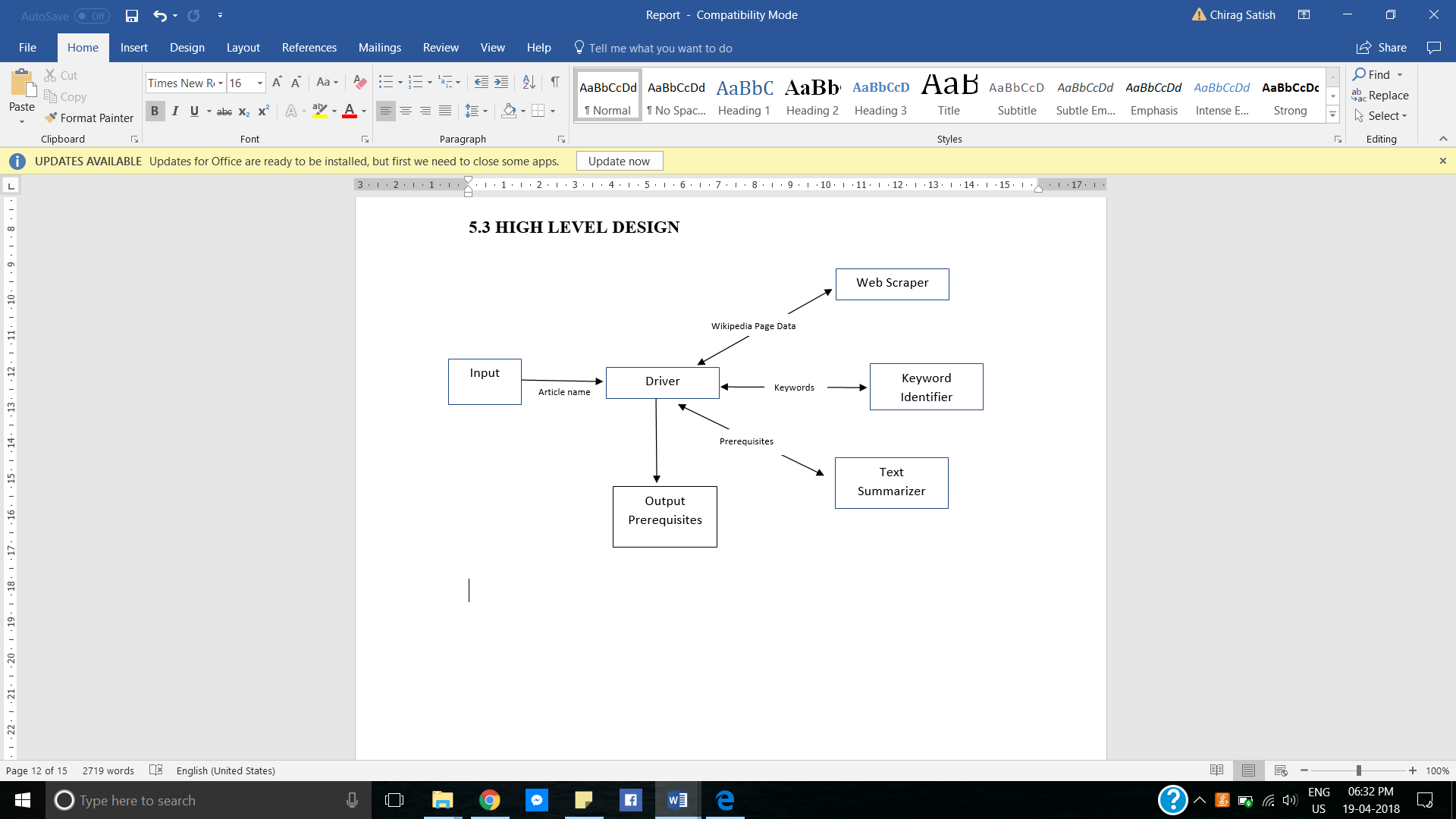
* Application Layer
* Liaison Layer
* Data Layer



**Fig 5.1 - System Architecture**

The data layer holds the corpus, or the data extracted from the Wikipedia pages. This is stored in a dictionary during the execution of the code. The liaison layer consists of the functions which are used to process the corpus. This includes web scraping, keyword identification and summarization modules. Web scraping module is used to extract the corpus from the Wikipedia pages. Keyword identification module identifies the keywords from this corpus. Summarization module summarizes the pages which are prerequisites to understand the current article. The front end is a command line interface used to take input from the user. The input is in the form of a string, here, the name of the personality.

**5.3 HIGH LEVEL DESIGN**

****

**Fig 5.2 Block diagram representing modules in the system**

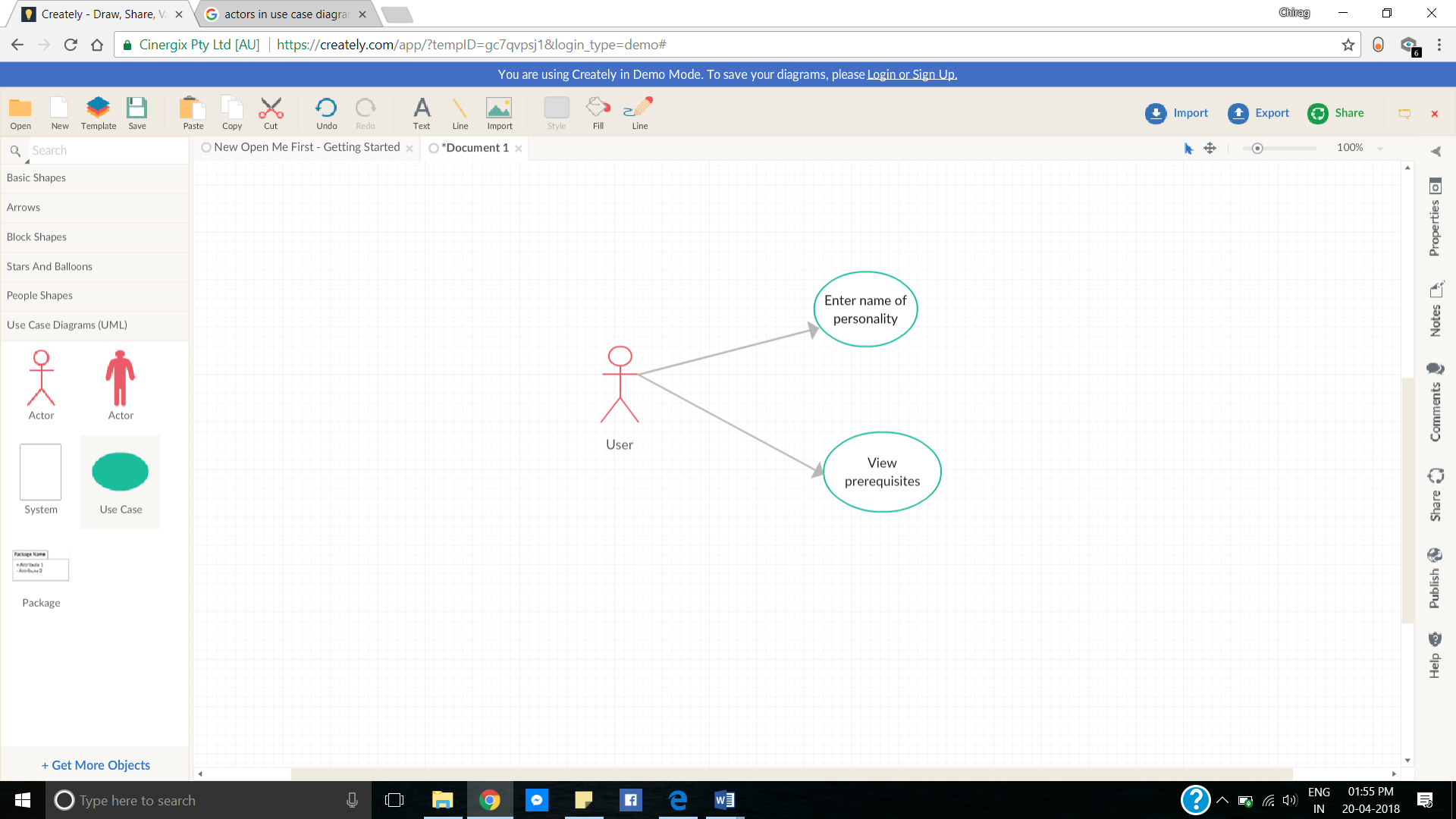
In high level design the modules that should be built for developing the system are identified. In other words, the foucus is on what modules are needed. Fig 5.2 shows the high level design along with the interconnections between modules.   
The user inputs the name of the Wikipedia for which the prerequisites are to be generated. The driver module forwards the name of article to the scraper module which returns the contents of the particular Wikipedia page. Next the keyword identifier processes this data to generate a list of keyphrases which are vital to understanding the page. Finally the text summarizer returns a succinct explanation of these keywords, which is then output to the user.

**CHAPTER 6**

**DETAILED SYSTEM DESIGN**

**6.1 USE-CASE DIAGRAM**

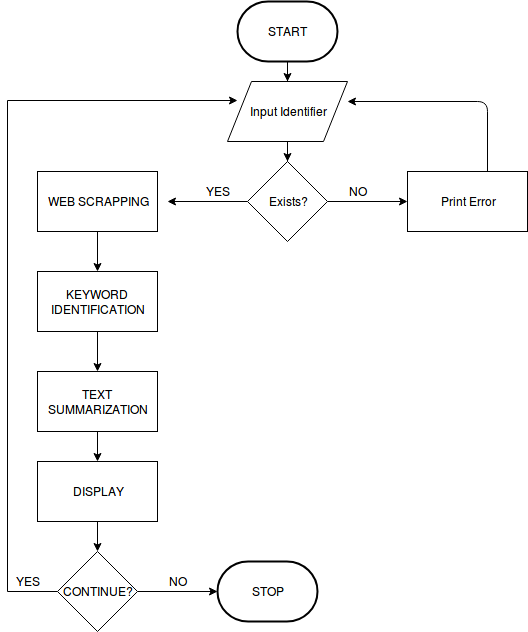
A use case diagram in the Unified Modelling Language (UML) is a type of behavioral diagram defined by and created from a use case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals, and any dependencies between those use cases. Due to their simplistic nature, use case diagrams can be a good communication tool for stakeholders. The drawings attempt to mimic the real world and provide a view for the stakeholder to understand how the system is going to be designed. [ ]



**Fig 6.1 USE-CASE diagram representing user activities**

As shown in figure 6.2, the user enters the name of the personality (Wikipedia article) he wishes to get pre-requisite information about. The system then returns this information which he can then view.

**6.2 ACTIVITY DIAGRAM**

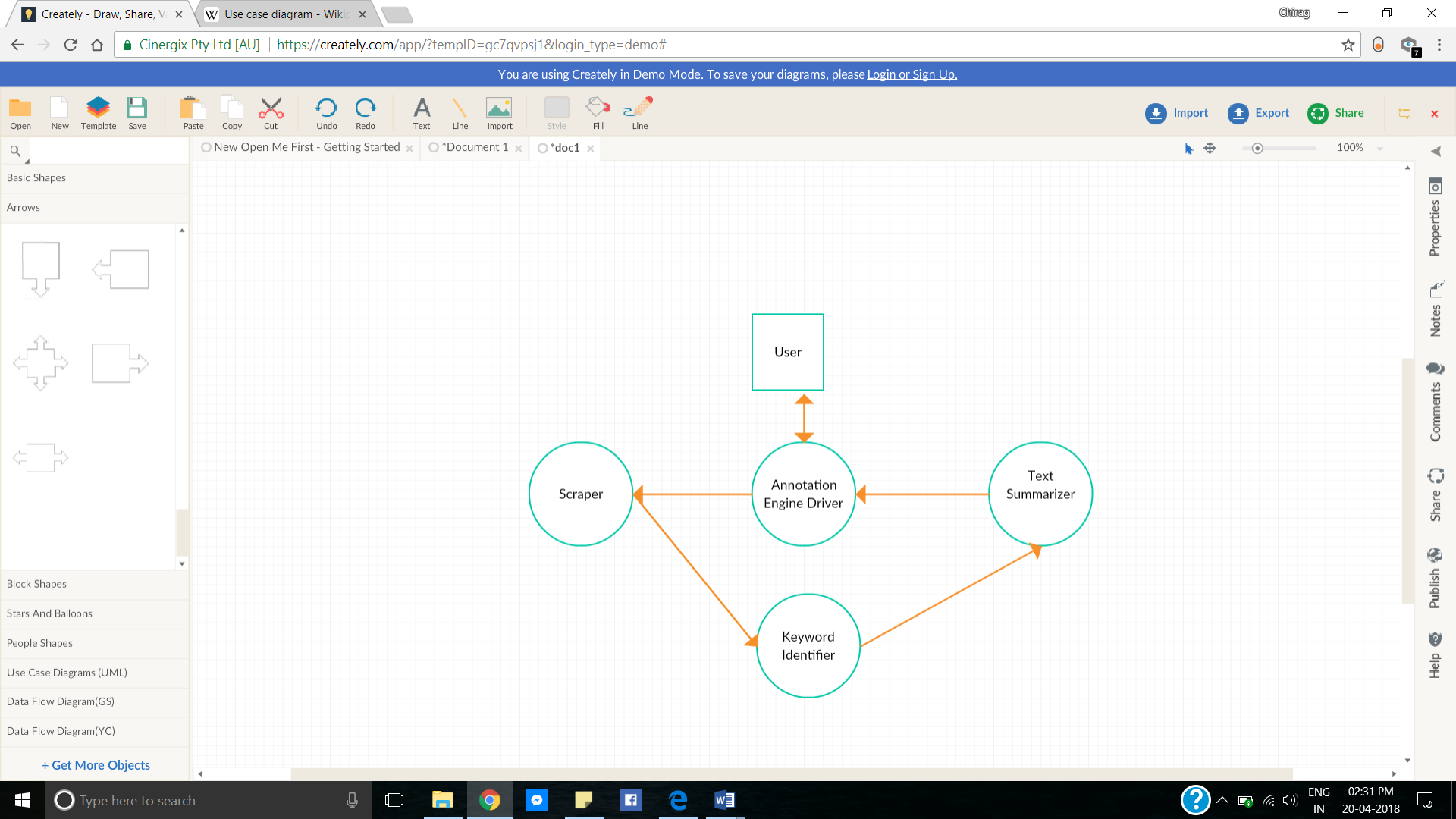
Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent or branched. It gives an idea of how the system functions when executed. [ ]  


**Fig 6.2 Activity Diagram of the system**

Figure 6.1 depicts the flow of activities in the annotation engine we have created along with conditional branches that occur throughout the various working scenarios.

**6.3 DATA FLOW DIAGRAM**

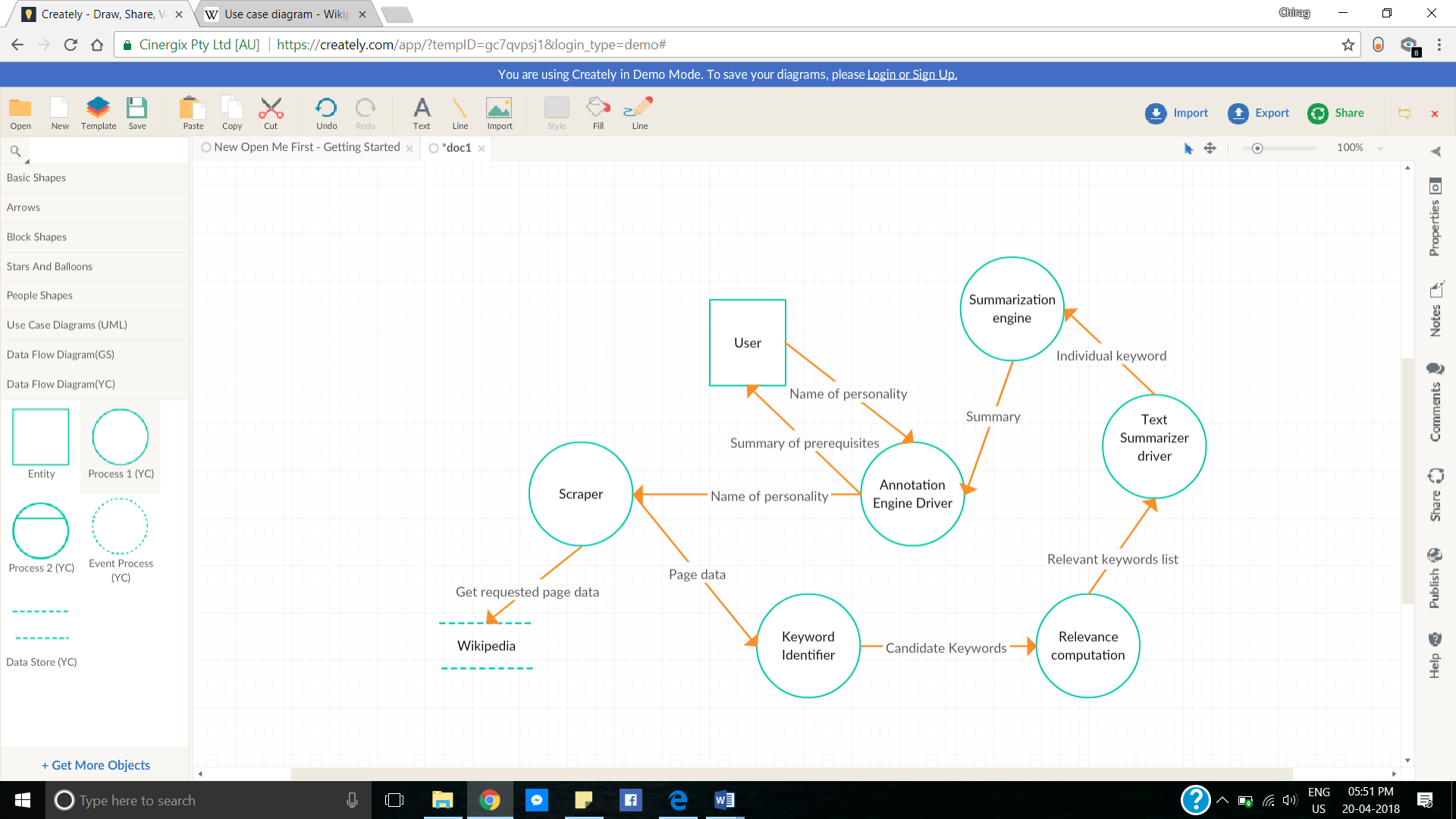
A data flow diagram (DFD) is a significant modelling technique for analyzing and constructing information processes. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a process model. [ ]



**Fig 6.3 DFD Level-0 (Context Diagram)**

A System Context Diagram (SCD) is a diagram that defines the boundary between the system, or part of a system, and its environment, presenting the entities that interact with it. This diagram is a high-level view of a system.

DFD is a designing tool used in the top-down approach to Systems Design. This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job and shows the flow of data between the various parts of the system. As we can see from figure 6.4, the entire system is decomposed into sub-modules which interact with data stores i.e. the Wikipedia repository.

****

**Fig 6.4 DFD Level-1**

**CONCLUSION**

Reading a new article on the web can be tricky, mainly due to the existence of numerous esoteric terms. Our system solves this problem by providing the reader a summarized version of these difficult terms which they may go through to understand the article better. With this, reading time for articles can greatly be reduced. We tested our system on Wikipedia articles on famous personalities, in which a combination of keyword extraction and text summarization techniques were implemented. Rapid Automatic Keyword Extraction (RAKE), proposed in [1] has been used to get all the keywords from the article, from which our algorithm picks the keywords requiring summarization.

The major drawback of our system is the execution time, which is preventing us from using it as a real-time Wikipedia tool. We believe that there is a lot of scope for such systems and with more research, an ideal version of the automatic annotation engine for webpages can be realized.

**FUTURE ENHANCEMENTS**

* Whenever a keyword is encountered, we visit that Wikipedia page and summarize it. Sometimes, even the summary contains arcane terms which require explanation. A system which can traverse through webpages till the summary is easily understandable can be developed as an extension of our system.
* Developing a Wikipedia add-on tool that can provide summary of prerequisites as soon as the page loads on the browser.
* Optimizing the algorithm to work effectively when internet connection is poor.
* Extending the system to other literature such as technical papers and web articles.
* To implement user profiling, to present summary according to the user. For example, a software engineer will be presented with an in-depth summary when they visit articles related to computer science whereas a simple summary will be presented to users from different fields of expertise.

**REFERENCES**

1. “Automatic Keyword extraction from Individual Documents” by Stuart Rose, David Engel, Nick Cramer
2. “Multi-document summarization using document set type classification” by Jun’ichi Fukumoto and Tomoya Sugimura
3. “Automatic text structuring and summarization” by Gerard Salton, Amit Singhal, Mandar Mitra and Chris Buckley
4. “A Survey on abstractive text summarization” by N. Moratanch, S. Chitrakala
5. “Automatic Extraction of Document Keyphrases for Use in Digital Libraries: Evaluation and Applications” by Steve Jones and Gordon W. Palmer
6. “Keyword extraction from a single document using word co-occurrence statistical information” by Matsuo Y and Ishizuka M
7. “Stop-words in keyphrase extraction problem” by S. Popova, L. Kovriguina, D. Mouromtsev, I. Khodyrev
8. BeautifulSoup - <https://pypi.org/project/beautifulsoup4/>
9. The Natural Language Toolkit - <https://pypi.org/project/nltk/>
10. NumPy - <http://www.numpy.org/>
11. Urllib - <https://docs.python.org/2/library/urllib.html>