Machine learning approach to word extraction

CHAPTER-1

INTRODUCTION

Today in this world of the internet to facilitate the task of reading and searching information has been necessary to find a way to reduce the size of a document without affecting the context and also highlight the keywords in the reduced document. The human unable to manually handle large text volumes must find an automatic analysis method adapted to the automatic processing of personal data. It is important to highlight the important and the difficult words to make the user better understand the document. Keyword or vital words in documents are set of significant words in a document that give high-level descriptions of the content for investigating readers and are useful tools for many purposes. They are used in academic articles to give an insight about the article so that the readers can determine whether the article is in their area if interest. In a textbook they are useful for the readers to identity the main points in their mind about a particular section. They can also be used for search engines in order to return more precise results in a shorter time. Since keywords describe the main points of a text, they can be used as a measure of similarity for text categorization. In summary, keywords are useful tools for scanning large amounts of documents in a short time. Despite the usefulness of the keywords very few of the current documents include them. In fact, many authors are not intend to extract word, or keywords and do not denote them unless they are not explicitly instructed to do so. Extracting words, keyword manually is an extremely difficult and time consuming process; therefore it is almost impossible to extract keywords manually even for the article published in a single conference. Therefore there is a need for an automated process that extracts keywords from documents.

Today, very large amount of information is available in online documents. As part of the effort to better organize this information for users, researchers have been actively investigating the problem of automatic text and word categorization. The bulk of such work has focused on topical categorization, attempting to sort documents according to their subject matter and behavior. In recent years there have been rapid growths in online discussion groups and review sites where a crucial characteristic of the posted articles is their sentiment or overall opinion towards the subject matter. A more related area of research is that of determining the genre of texts, subjective genres, are often one of the possible categories. But, while techniques for genre categorization and subjectivity detection can help us recognize documents that express An opinion they do not address our specific classification test of determining what that opinion actually is. Most previous research on sentiment-based classification has been at least partially knowledge-based. Some of this work focuses on classifying the semantic orientation of individual words or phrases, using linguistic heuristics or a pre-selected set of seed words. Here we choose to work with the domain of experimentally convenient because there are large online collections of such data because those data often summarize their overall sentiment with a machine-extractable rating indicator hence we did not need to hand label the data for supervised learning or evaluation purposes.

CHAPTER-2 LITERATURE SURVEY

2.1.1 Research Papers on Existing System:

1. ***"Personalized Text Content Summarizer for Mobile Learning: An Automatic Text Summarization System with Relevance Based Language Model"*** 2012 IEEE

Fourth International Conference on Technology for Education Doi: 10.1109/T4E.2012.23

Authors: Guangbing Yang; Dunwei Wen; Kinshuk; Nian-Shing Chen;Erkki Sutinen

Description:

With the rapid growth of mobile technology, mobile learning (m-learning) is rapidly becoming one of the mainstreams in today's e-learning. The fast evolution of mobile technology provides great potential to improve m-learning performance. Meanwhile, this swift development makes it more possible for mobile learners to access "on-the-fly" information. However, with millions of data available online in different forms or media types, learners find it extremely difficult to extract useful information, especially for learning purposes. In addition to this oft-decried information overloading problem, the small display screen, and limited network. The bandwidth and storage capability of mobile devices also have the potential to impede the development of the m- m-learning. For example, showing a large amount of text on a small screen of a mobile device requires scrolling, which is not a user-friendly task on mobile devices; also, delivering a big chunk of data across the mobile network can cost more and slow down the transaction speed significantly. Moreover, m-learners are usually willing to study using their mobile devices during their commutes. They expect to assimilate the information in a very limited time. In this case, it would be helpful if they could obtain the condensed content and important points rather than the entire learning content. Since both situations are content-related, if the content size could be reduced somehow, both problems might be alleviated. However, reducing the content may have a negative impact on the understanding of the meaning conveyed by it. Also, different learners may have different perspectives to the same learning content based on their learning preferences, interest, and prior knowledge.

2. "Machine Learning for Sentiment Analysis on the Experience Project"

Authors: Raymond Hsu; Bozhi See; Alan Wu

Description:

They aimed to extract the human emotions from the text. They have employed various machine learning concepts such as latent dirichlet allocation algorithm, word net synsets etc. to predict reader reaction from projects. They concentrated on the scenarios like; one might ask what the difficulty of the two tasks is and what level of accuracy would be considered successful. To answer the question of how hard the two tasks are, we can compare one's system performance against that of humans. They conducted a scaled-down version of the experiment where they had humans attempt the same two classification tasks as our models. Performance at the human level is often considered the target goal in sentiment analysis. Notices that they are not asking humans what are their reactions; they are asking them to predict what they think other people would have voted on.

3.*" Survey of Keyword Extraction Techniques"* 2014, IEEE ***Author: Brian Lott***

Description:

Keywords are commonly used for search engines and document databases to locate information and determine if two pieces of test are related to each other. Reading and summarizing the contents of large entries of text into a small set of topics is difficult and time consuming for a human, so much so that it becomes nearly impossible to accomplish with limited manpower as the size of the information grows. As a result, automated systems are being more commonly used to do this task.

4." Optimal Features Set For Extractive Automatic Text Summarization"2015

Fifth International Conference on Advanced Computing & Communication Technologies

Doi: 10.1109/ACCT.2015.123 **Authors:** Yogesh Kumar Meena;Peeyush Deolia; Dinesh Gopalani

Description:

The goal of sentiment analysis is to extract human emotions from text. This paper applies various machine learning algorithms to predict reader reactions to excerpts from the Experience Project. Metrics such as accuracy of prediction and precision/recall are presented to gauge the success of these different algorithms. We propose a system to process the documents and to predict human reactions, as well as provide results. We discuss various methods and their advantages and disadvantages in sentiment analysis for these documents. Finally, we comment on applying our finding to sentiment analysis in a more general sense. One application of machine learning is in sentiment analysis. In this field, computer programs attempt to predict the emotional content or opinions of a collection of articles. This becomes useful for organizing data, such as finding positive and negative reviews while diminishing the need for human effort to classify the information.

5." Thumbs up? Sentiment Classification using Machine Learning Techniques" Proceedings of the Conference on Empirical Methods in Natural Association for Computational Linguistics. Language Processing (EMNLP), Philadelphia, July 2002 **Authors:** Bo Pang; Lillian Lee; Shivakumar Vaithyanathan

Description:

Today, very large amounts of information are available in online documents. As part of the effort to better organize this information for users, researchers have been actively investigating the problem of automatic text categorization. The bulk of such work has focused on topical Categorization, attempting to sort documents according to their subject matter (e.g., sports vs. politics). However, recent years have seen rapid growth in online discussion groups and review sites where a crucial characteristic of the posted articles is their sentiment or overall opinion towards the subject matter| for example, whether a product review is positive or negative. Labeling these articles with their sentiment would provide succinct summaries to readers; indeed, these labels are part of the appeal and value-add of such sites as www.rottentomatoes.com, which both labels movie reviews that do not contain explicit rating indicators and normalizes the different rating schemes that individual reviewers use. Sentiment classification would also be helpful in business intelligence applications and recommender systems where user input and feedback could be quickly summarized; indeed, in general, freeform survey responses given in natural language format could be processed using sentiment categorization. Moreover, there are also potential applications to message filtering; for example, one might be able to use sentiment information to recognize and discard flames.

6."Automatic Text Summarization of Wikipedia Articles" 2015 International Conference on Communication, Information & Computing Technology (ICCICT)

Doi: 978-1-4799-5522-0/15/ **Authors:** Dharmendra Hingu; Deep Shah; Sandeep S. Udmale

Description:

Natural languages can be broadly defined as any informal language that can be learnt by any person. They differ from formal languages like computer programming languages which have a proper structure and syntax. Natural Language Processing (NLP) is the understanding of the context of these natural languages and/or generating a text in natural language. Radev et al. defines summary as "a text that is produced from one or more texts, that conveys important information in the original text, and that is no longer than half of the original text and usually significantly less than that. The objective of automatic text summarization is the reduction of a given text to a smaller number of sentences without leaving out the main ideas of the original text.

The architecture of a summarization system consists of interpretation, transformation and generation. Text summarization is broadly classified into four categories. The first category is a classical approach which relies on a selection of salient information based on features like frequency, cue phrases, title and heading words, and sentence location. The second category is a corpus-based approach which uses the concept of statistics and then decides the appropriate combination of features mentioned in classical approach. The third category is an exploiting discourse structure which exploits models of discourse structure, while maintaining a relatively domain-independent and knowledge poor approach. The final category is knowledge rich approach which models the rich knowledge requirements for summarization in a particular domain. In this category, a relatively rich structured representation has been built, which the summarization process used as an input. First three categories focus mainly on interpretation but final category focuses on transformation and generation. Microsoft word 2007 uses the summarization algorithm which identifies word frequency in the given documents and a percentage of higher scored sentences are displayed in the summary.

CHAPTER-3

SYSTEM ANALYSIS

3.1. Existing system and its drawbacks:

The existing system basically involves in extracting sentences from given text document and based on the extraction algorithm it does summarization part and provides the summarized output. It only extracts the sentence in the document, but not the highlighted words which are important in the given document. A system which is restricted in providing only about the summary but not the clear understanding in terms of meaning. It is very expensive for an external user. Also, the behaviors of the text and words extracted have failed in providing maximum satisfaction to the customers. Limitations of the existing system is that they have no segmentation, less user satisfaction, are not accurate in their final results, are less reliable, and fail to extract proper behavior of the text extracted.

3.2. Proposed System:

This proposed system helps the user by extracting the important keywords automatically. It also helps the user to know the meaning of that particular word without the need of searching separately. Whenever a new keyword appears in a document, the software automatically detects it and adds the word to its preloaded keyword. This makes the machine learn by itself without any manual intervention. Hence this method helps in better understanding of the document with lots of information. This process involves extracting important words, and phrases from a given text document and providing the meaning of the highlighted word for a better understanding of the document. The heart of this system is a sentimental analysis of text and words extracted from the given document

3.2.1. Advantages of the proposed system:

- Provide security for the users and also file access permission is provided only for authenticated users hence file cannot be retrieved by intruders and files will not be damaged.
- Provide Topic categorization: the user will be given with two types of documents to upload.
- In each category like the educational category we have subcategories: science, engineering fields, social studies, and English literature. In the news category we have subcategories: politics, current affairs, health care, transport, tourism, entertainment, sports.
- After uploading the document the content will be summarized, and word extraction and behavior analysis will be done. Also, that particular document will be stored in that respective category so that the other users can read the information by accessing that category of their choice
- We also provide save documents option, which it contains user's summarized documents and the document uploaded in any file category will be exclusively seen here.
- The separate users category is also provided where user can upload any type of document and get summarized text, word extracted and behavior of the text been saved in their profile.

CHAPTER-4 *SYSTEM REQUIREMENTS*

4.1. Hardware requirements:

•	Processor:	Pentium IV onwards
•	Hard Disk Space:	Minimum of 512MB.
•	Speed:	1.2 GHz+
•	Mobile:	Android Mobile.
•	App. Version:	Kit Kat or above.

4.2. Software requirements:

•	Operating System:	Windows XP or higher.
•	Language:	Java, Python, HTML, CSS, XML
•	Java Software:	JDK 1.7 or above, SDK.
•	Tool:	Android Studio,Jython
•	Database:	SQLite. MySQL

CHAPTER-5 SYSTEM DESIGN

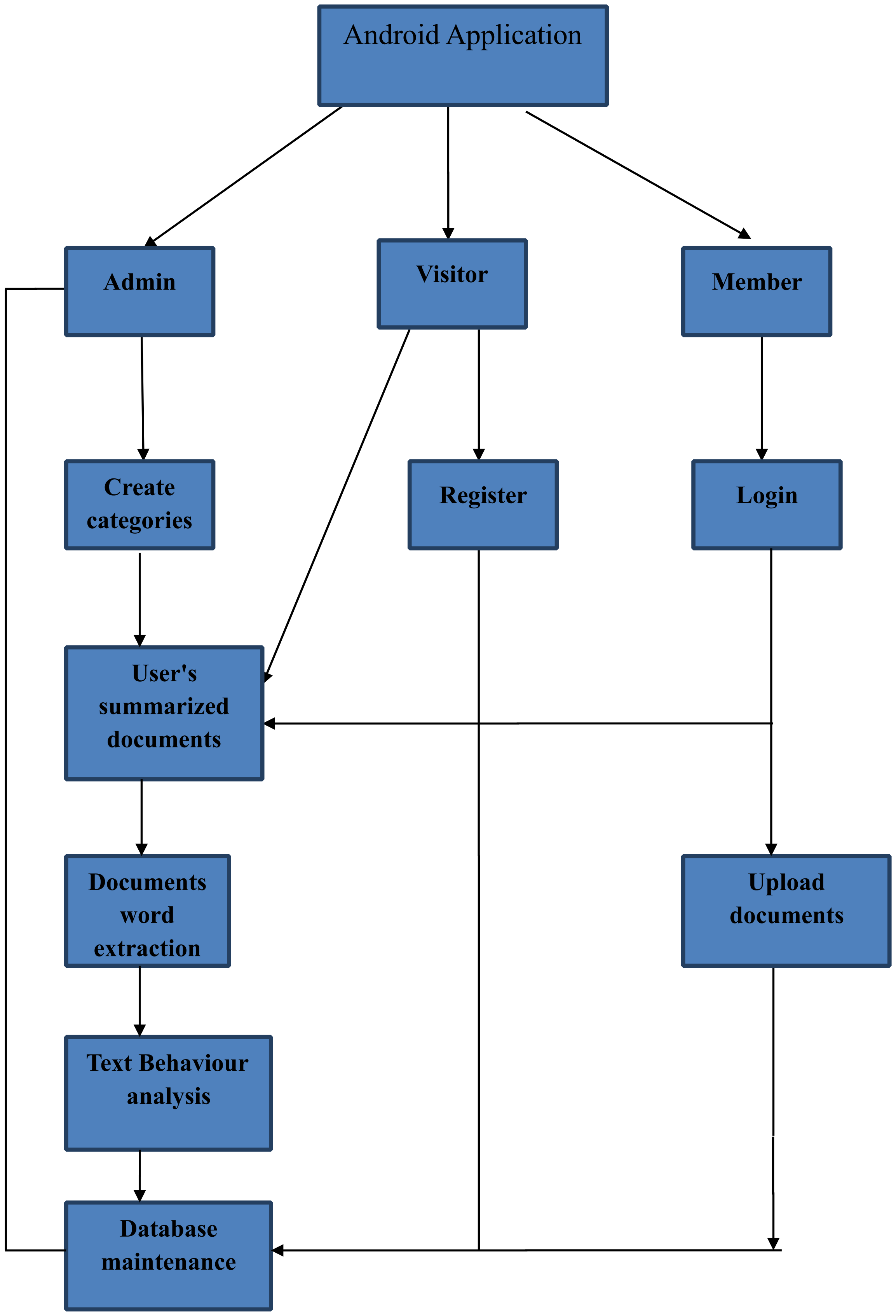
5.1. Introduction:

This chapter gives over all flow of the project and algorithm used in the design. Implementation is a stage in software development where the software design is realised as a set of program units. The objects that are identified in the design stage are implemented.

System design is a blue print of the solution for a system. Design of the system can be defined as the process of applying various techniques and principles for the purpose of defining a process are a system in sufficient details to permit its physical realisation. System design is concerned with how the system functionalities have to be provided by the different components of the system. Thus system design is a "haw to" approach to the creation of a new system. The design of the system is perhaps the most critical factor affecting the quality of the software; it has a major impact on the later phases particularly testing and maintenance. The design activity often results in three separate outputs:-

- **Architecture.**
- **High level design.**
- **Detailed design**.

5.2 Architecture: Fig 1: Architecture Diagram



5.2.1. Three-Tier Architecture:

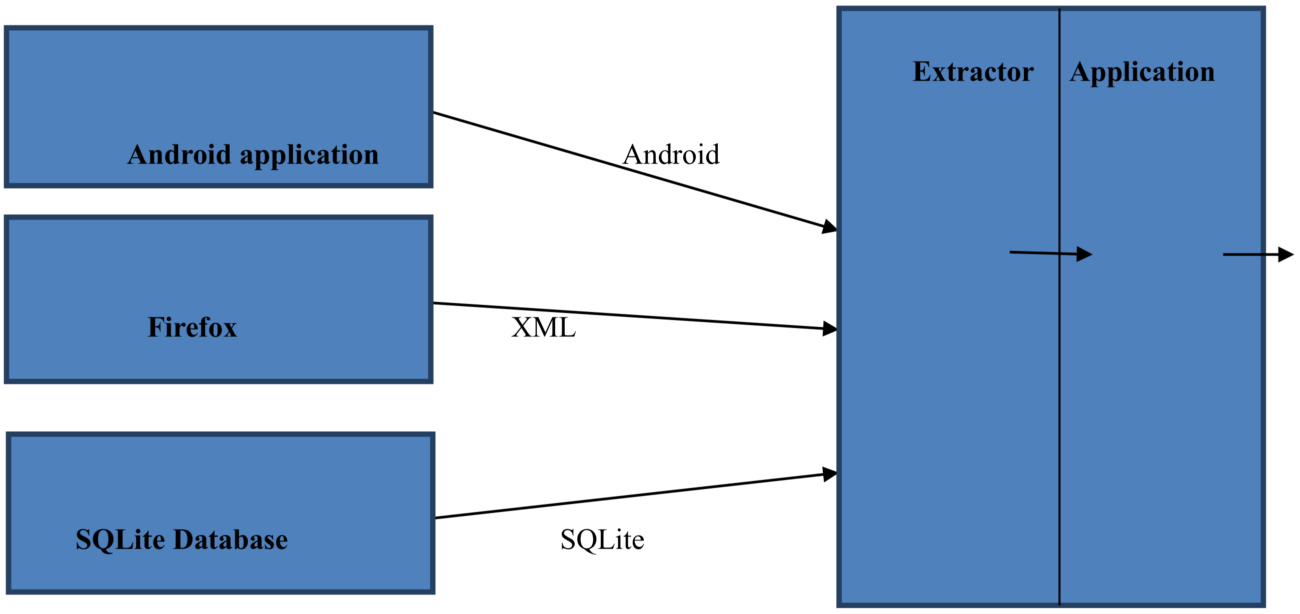


Fig 2: ***Three-Tier Architecture***

5.3 High-Level Design:

In high-level design the modules that should be built for developing the system and the specification of these modules. At the end of the system design all major data structures, file formats, output formats, etc. Are also fixed. The focus is on identifying the modules. In other words, the attention is on what modules are needed.

5.4. Detailed Design:

In the detailed design, the internal logic of each of the modules is specified. The focus is on designing the logic for each of the modules. In other words, how modules can be implemented in software is the issue. A design methodology is a systematic approach to creating a design of application. Most methodology focuses on high-level design.

CHAPTER-6 *SYSTEM DETAILED 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.

The main purpose of a use case diagram is to show what system functions are performed for which actors. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If the interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined.

6.1.1 Use cases:

A use case describes the sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

6.1.2 Actors:

An actor is a person, organization or external system that plays a role in one or more interactions with the system.

6.1.3 System Boundary Boxes:

A rectangle is drawn around the use cases called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not.

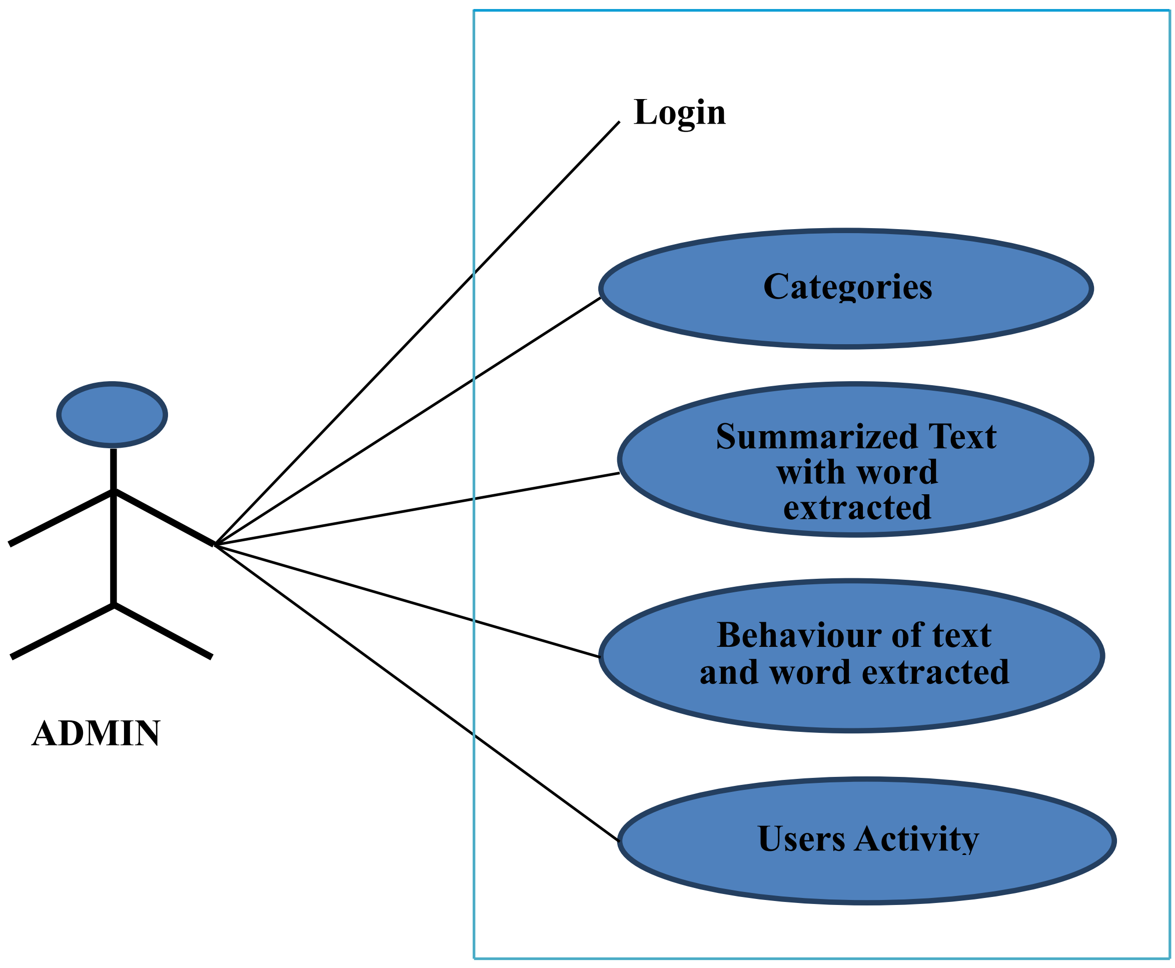

Login
Categories
Summarized Text with word extracted
Behaviour of text and word extracted
Users Activity
ADMIN

Fig 3: USE-CASE representing the admin activities

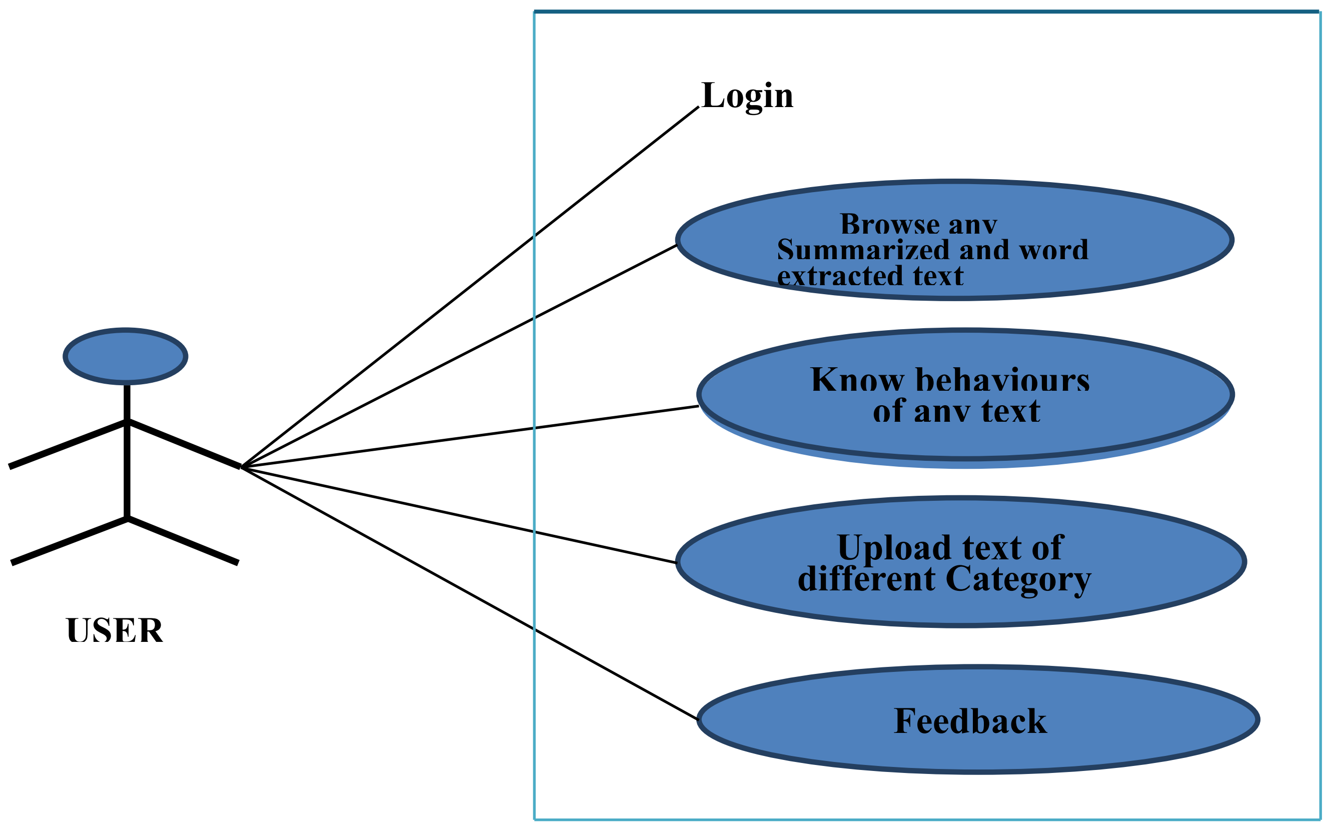

Login
Browse anv Summarized and word extracted text
Know behaviours of anv text
Upload text of different Category
Feedback
USER

Fig 4: *USE-CASE to represent the activities of the user*

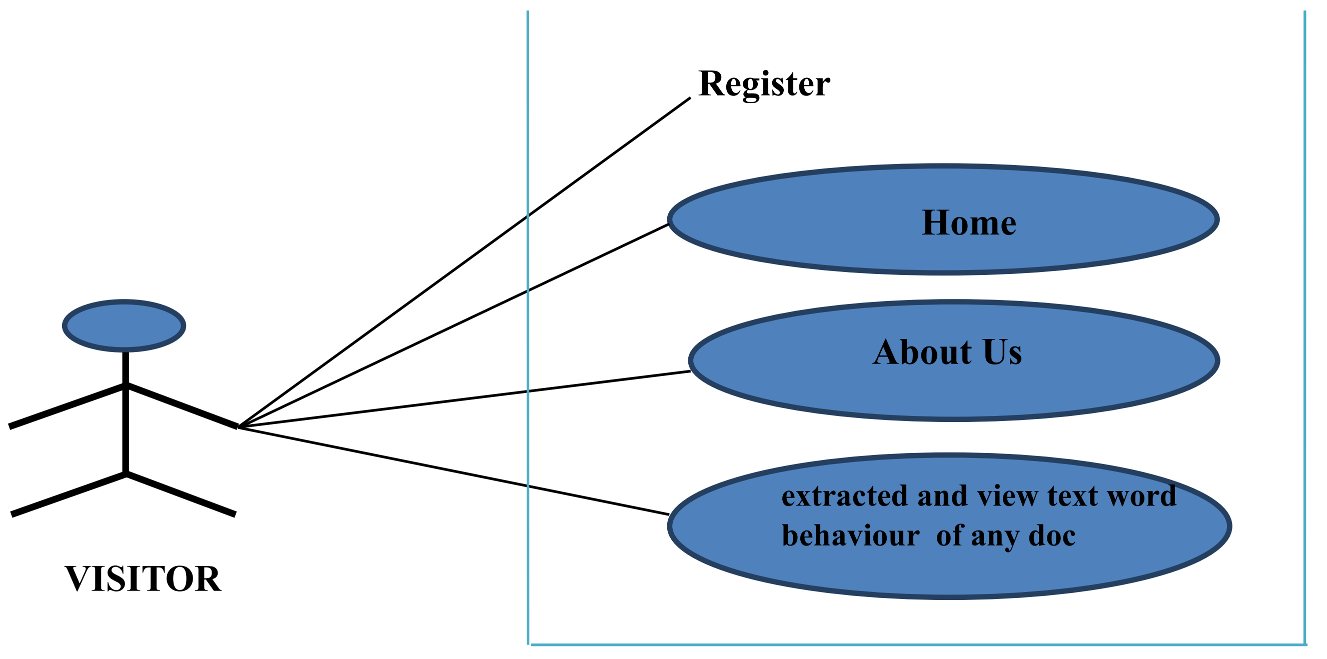


Fig 5: USE-CASE to represent the activities of the Visits

6.2. Sequence Diagram:

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

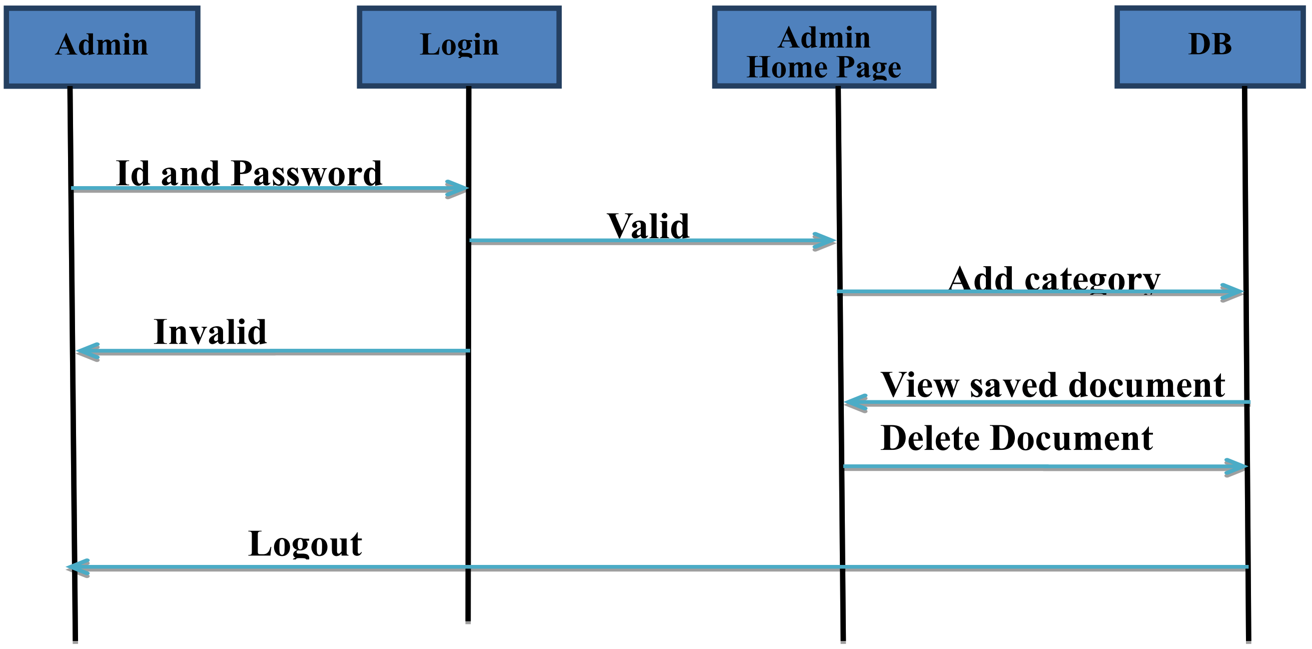


Fig 6: ***Admin's sequence diagram***

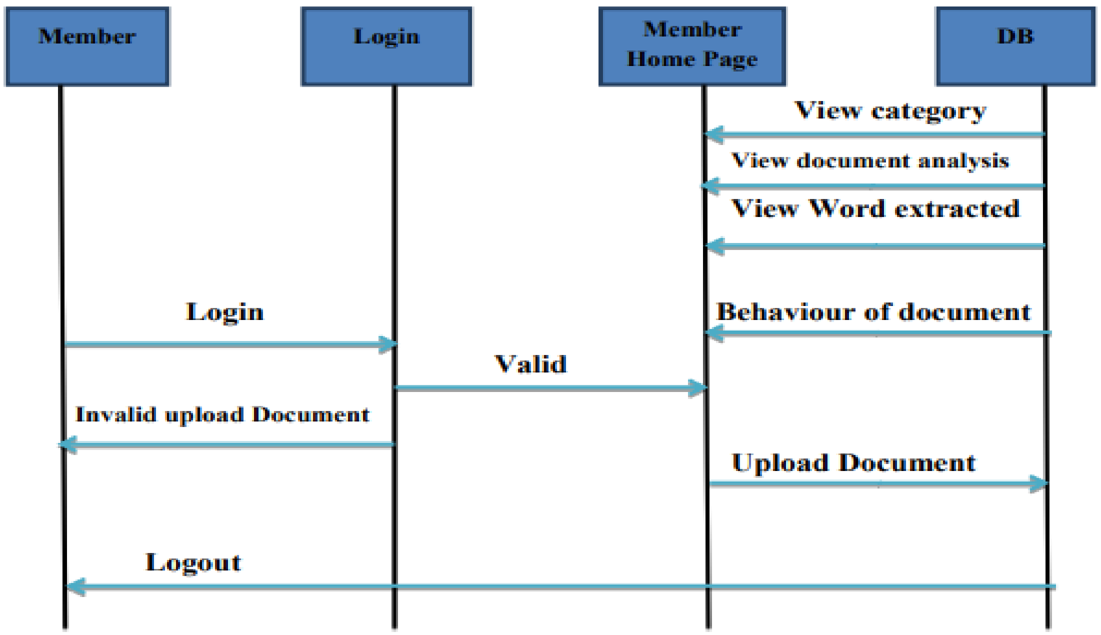


Fig 7: ***Members' sequence diagram***

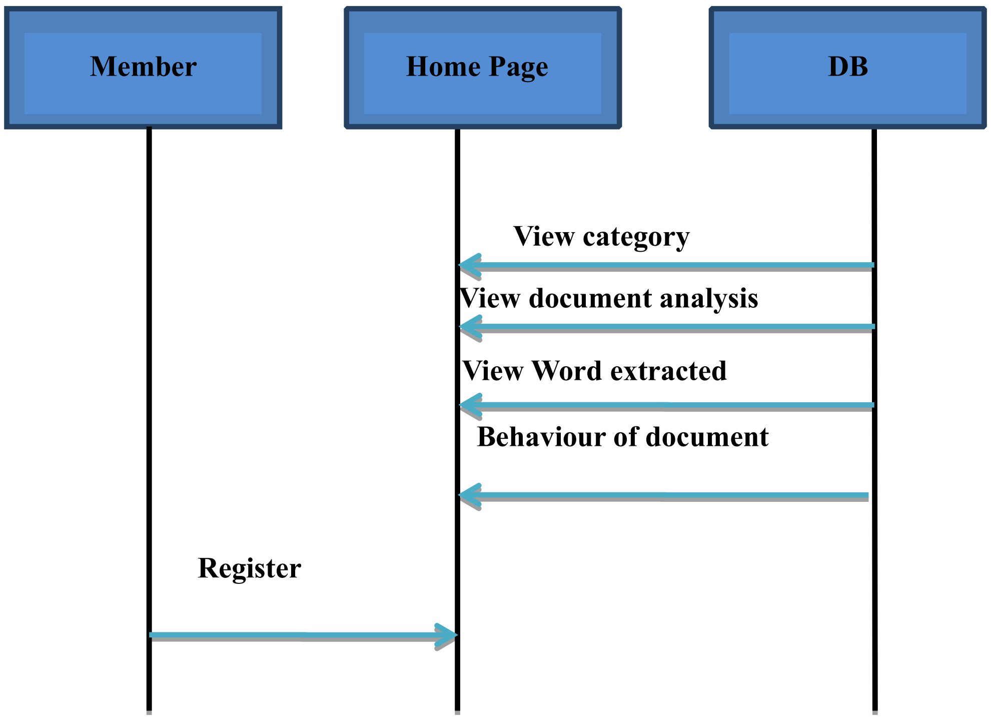


Fig 8: ***Visitor's sequence diagram***

6.3 Data Flow Diagram:

A data flow diagram (DFD) is a significant modelling technique for analysing and constructing information processes.DFD literally means an illustration that explains the course or movement of information in a process. 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.

A DFD can be utilized to visualize data processing or a structured design. A DFD illustrates technical or business processes with the help of the external data stored, the data flowing from a process to another, and the results.

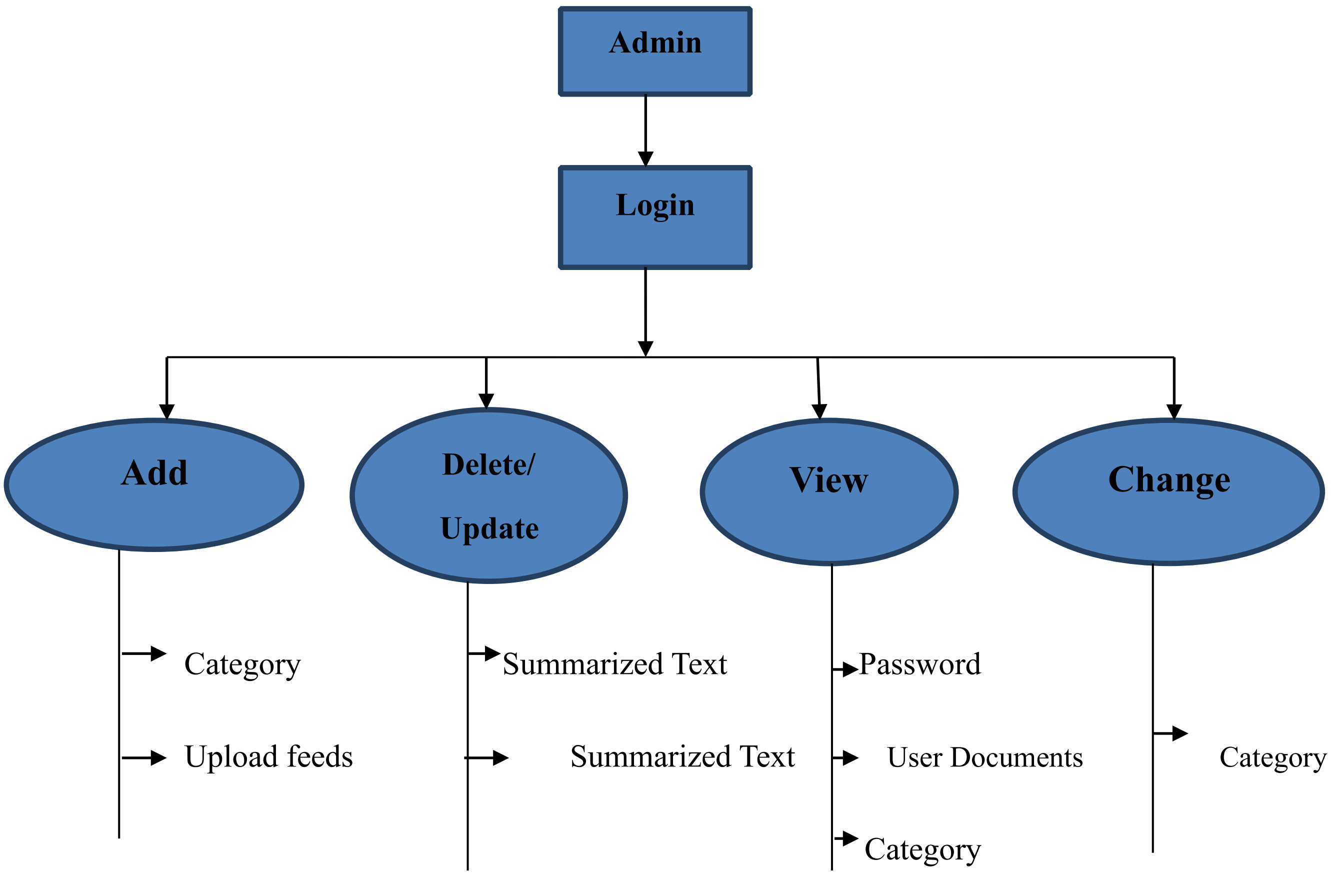


Fig 9: ***Admins DFD***

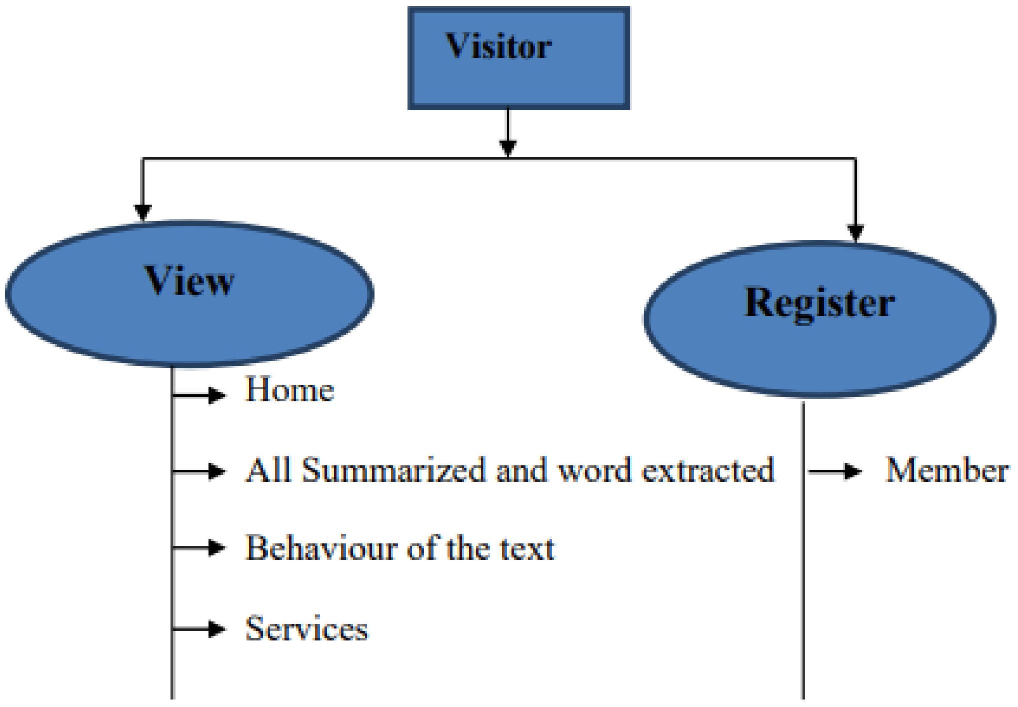


Fig 10: Visitor's DFD

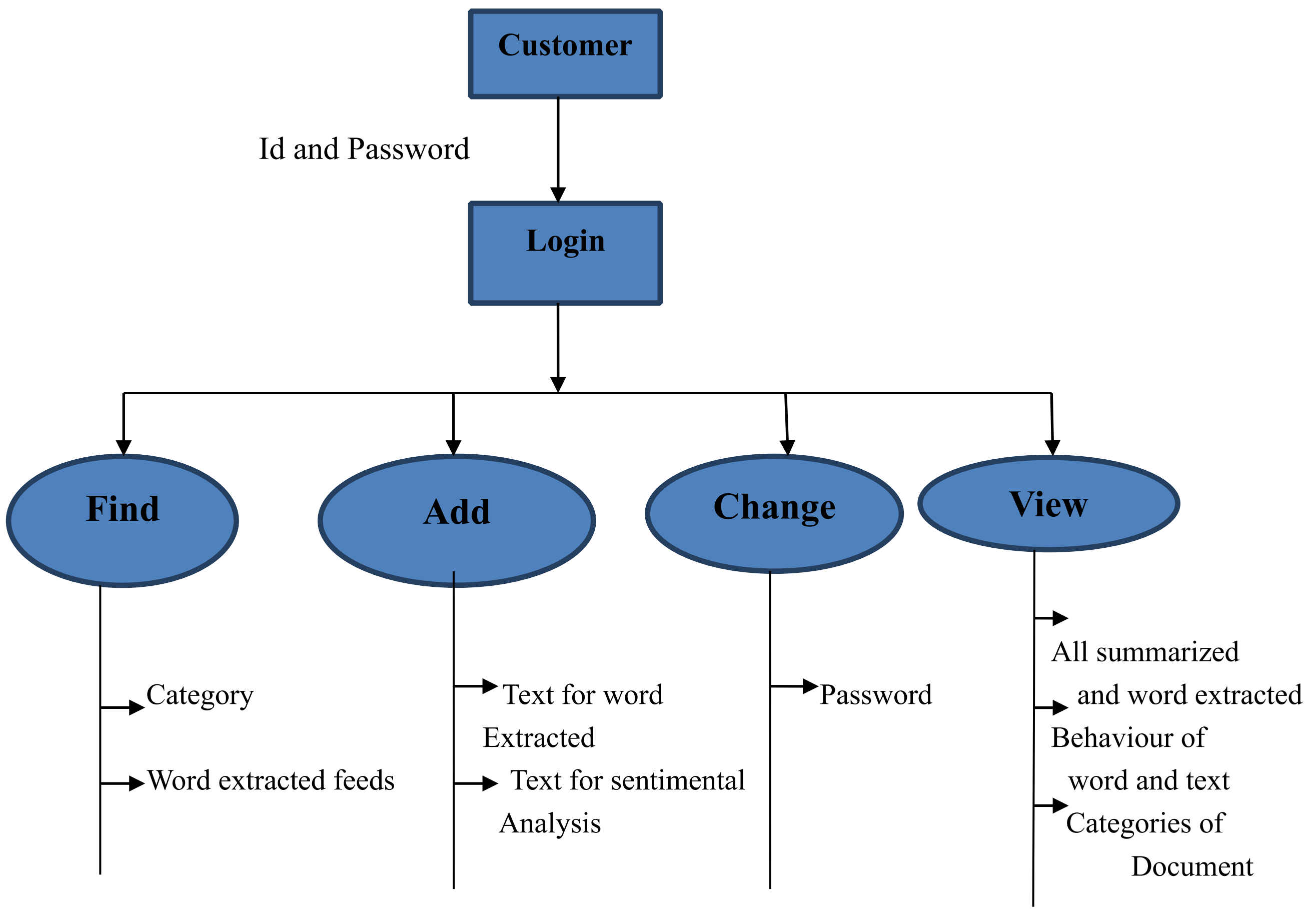


Fig 11: ***Member's DFD***

CHAPTER-7

IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on new system for the users, that it will work efficient and effectively.

It involves careful planning, investigating the current system, and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods.

The implementation process started with preparing a plan for the implementation of the system. According to this plan, discussion has been made regarding the equipment, resources and test activities to be performed. Thus, a clear plan was prepared for activities.

7.1 Module Implementation and related concepts:

7.1.1 Anaconda:

Anaconda is a Python distribution that is particularly popular for data analysis and scientific computing Open source project developed by Continuum Analytics, Inc. Available for Windows, Mac OS X and Linux Includes many popular packages: NumPy, SciPy, Matplotlib, Pandas, IPython, Cython Includes Spyder, a Python development environment Includes conda, a platform-independent package manager.

7.1.2 Spyder 3.5:

Spyder formerly known as **Pydee** is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates NumPy, SciPy, Matplotlib and IPython, as well as other open source software. It is released under the MIT license.

Spyder is extensible with plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows with WinPython and Python(x,y), on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu. Spyder makes use of Qt either through the binding PyQt or PySide. This flexibility is reached through a small abstraction layer called QtPy.

7.1.3 Android SDK:

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in Java programming language using the Android Software Development Kit (SDK), but other development environments are also available.

Android provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment. The documents listed in the left navigation provide details about how to build apps using Android's various APIs provides multiple entry points for a single app and allows any app to behave as a user's "default" for an action that other apps may invoke.

7.1.4 Python:

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

7.1.5 Flask:

Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. It is BSD licensed.

Flask is called a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, and upload handling, various open authentication technologies, and several common framework-related tools. Extensions are updated far more regularly than the core Flask program.

7.2 Steps involved in Implementation:

7.2.1 Machine learning:

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\Java\\jre7') # for 32-bit version
library(rJava) require("openNLP") require("NLP")

s <- paste("Complimentary gym access for two for the length of stay $12 value per person per day")

tagPOS <-  function(x, ...) {   s <- as.String(x)     word_token_annotator
<- Maxent_Word_Token_Annotator()     a2 <- Annotation(1L, "sentence",
1L, nchar(s))     a2 <- annotate(s, word_token_annotator, a2)
   a3 <- annotate(s, Maxent_POS_Tag_Annotator(), a2)     a3w <-
a3[a3$type == "word"]
   POStags <- unlist(lapply(a3w$features, `[[`, "POS"))
   POStagged <- paste(sprintf("%s/%s", s[a3w], POStags), collapse = " ")     list(POStagged =
POStagged, POStags = POStags)
 }
 tagged_str <-  tagPOS(s)   tagged_str

#$POStagged
#[1]  "Complimentary/NNP gym/NN access/NN for/IN two/CD for/IN the/DT length/NN of/IN stay/NN $/$
12/CD value/NN per/IN     person/NN per/IN day/NN" #
#$POStags
#[1] "NNP" "NN"  "NN"  "IN"  "CD"  "IN"  "DT"  "NN"  "IN"  "NN"  "$"   "CD"
```

7.2.2 Tokenization:

```
class Splitter(object):

   def __init__(self):
      self.nltk_splitter = nltk.data.load('tokenizers/punkt/english.pickle')        self.nltk_tokenizer =
nltk.tokenize.TreebankWordTokenizer()

   def split(self, text):
      """ input format: a paragraph of text        output format: a list of lists of words. e.g.: [['this', 'is', 'a',
'sentence'], ['this', 'is', 'another', 'one']]        """        sentences = self.nltk_splitter.tokenize(text)
tokenized_sentences = [self.nltk_tokenizer.tokenize(sent) for sent in sentences]        return
tokenized_sentences
```

7.2.3 Data set dictionary tagger:

```
def value_of(sentiment):
   if sentiment == 'positive': return 1     if
sentiment == 'negative': return -1
   if sentiment == 'inc': return 5     return 0

def sentence_score(sentence_tokens, previous_token, acum_score):         if not
sentence_tokens:         return acum_score     else:
       current_token = sentence_tokens[0]         tags =
current_token[2]         token_score = sum([value_of(tag) for tag
in tags])         if previous_token is not None:
          previous_tags = previous_token[2]
if 'inc' in previous_tags:                 token_score *=
2.0             elif 'dec' in previous_tags:
              token_score /= 2.0             elif
'inv' in previous_tags:
token_score *= -1.0
       return sentence_score(sentence_tokens[1:], current_token, acum_score + token_score)

def sentiment_score(review):
   return sum([sentence_score(sentence, None, 0.0) for sentence in review]) def usage():
   """ This function prints the usage information"""     import sys

   sys.exit(1)

def main():     import
sys
   # check whether no commandline argument has been entered and show usage     if
len(sys.argv) == 1:
       usage()     else:
       # read from stdin if "-" is present as first commandline argument         if
sys.argv[1] == "-":             text = sys.stdin.read()         else:
          # open file in any other case; should there be errors, print error message and usage             try:
            file = sys.argv[1]
fd = open(file, 'r')                 text =
fd.read()                fd.close()
except:
            print ("ERROR: Input file could not be opened.")                usage()
splitter = Splitter()     postagger = POSTagger()     dicttagger = DictionaryTagger([
'dicts/positive.yml', 'dicts/negative.yml',
                           'dicts/inc.yml', 'dicts/dec.yml', 'dicts/inv.yml'])

   splitted_sentences = splitter.split(text)     pprint(splitted_sentences)

   pos_tagged_sentences = postagger.pos_tag(splitted_sentences)     pprint(pos_tagged_sentences)

   dict_tagged_sentences = dicttagger.tag(pos_tagged_sentences)     pprint(dict_tagged_sentences)

   print("analyzing sentiment...")
   score = sentiment_score(dict_tagged_sentences)     if
score<0:
      print("Document contents Negativity")     elif
score==0:
      print("Document nutral")     elif score>0:
print("Document contents positivity")
print(score)

if __name__ == "__main__":    main()
```

7.2.4. Removal of unwanted info:

```
Import sys
   # check whether no commandline argument has been entered and show usage     If len
(sys.argv) == 1:         Usage ()     else:
      # read from stdin if "-" is present as first commandline argument         if
sys.argv[1] == "-":            text = sys.stdin.read()         else:
         # open file in any other case; should there be errors, print error message and usage            try:
            file = sys.argv[1]
fd = open(file, 'r')               text =
fd.read()                fd.close()
except:
            print ("ERROR: Input file could not be opened.")
```

CHAPTER-8

TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.[1] Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects).

It involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

8.1 Purpose of Testing:

There are two fundamental purposes of testing: Verifying procurement specifications and Managing Risk. First, testing is about verifying that what was specified is what was delivered: it verifies that the product (system) meets the functional, performance, design and implementation requirements identified in the procurement specifications. Second, testing is about, managing risks for the acquiring agency and the system vendor/ developer/ integrator. The testing programme is used to identify when the work has been completed so that the contract can be closed, the vendor paid, and the system shifted by the agency into the warranty and maintenance phase of the project.

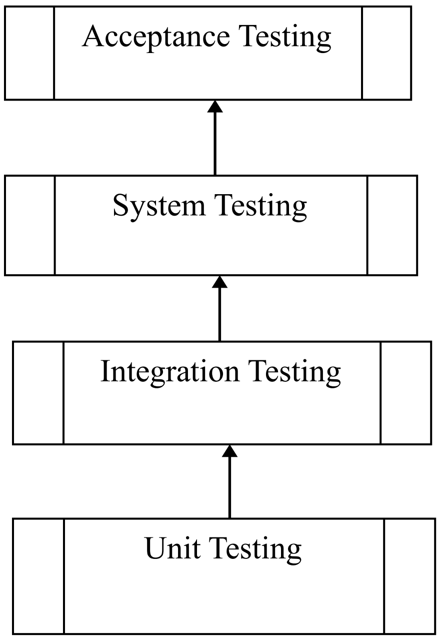
8.2 Black Box Testing:

The technique of testing without having any knowledge of interior working of the application is Black box testing. The tester is obvious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

8.3 White Box Testing:

White box testing is the detailed investigation of the internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look at the source code and find out which unit of the code is behaving inappropriately.

Fig 12: ***Different levels of testing***



8.4 Unit Testing:

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by developers to ensure that code meets its design and behaves as intended.

8.5 Integration Testing:

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

8.6 System testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

As a rule, system testing takes, as its input, all of the "integrated" software components that have passed testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

8.7 Acceptance Testing:

In engineering and its various sub disciplines, acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance.

In systems engineering it may involve black-box testing performed on system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

Software developers often distinguish acceptance testing by the system provider from acceptance testing by the customer (the user or client) prior to accepting transfer of ownership. In the case of software, acceptance testing performed by the customer is known as user acceptance testing (UAT), end-user testing, site (acceptance) testing, or field (acceptance) testing.

8.8 Test Cases:

TC#	Description	Expected Input	Expected Output	Status of execution
TC# 1	Software should perform in all platforms.	It should accept documents irrespective any OS.	Provide summary with proper alignment	Pass
TC# 2	Spacing full stops and other delimiters should be considered seriously.	Spaces are used as delimiters for words and full stops for sentences.	Proper summary of given document.	Pass
TC# 3	Document must have minimum of 50 words.	Document with more than ten sentences.	Important (heavy weighted) sentences.	Pass
TC# 4	Setting percentage for extraction of sentences.	User sets percentage for extraction of sentences.	User gets summary according to percentage.	Pass
TC# 5	Document should not be zipped or encrypted.	User should submit file in text form to get better result	Extracted sentences in text format.	Pass
TC# 6	To check whether document is junk	The document must contain useful and meaningful information.	The document is rejected if it is rejected.	Pass
TC# 7	Document should predominantly contain text images and equations must be as less as possible.	User must submit document which are not so dependent on images	The document containing only text and not images .	Pass
TC# 8	Document uploaded is analysed based on sentiment.	User must upload a sensible document.	Words are displayed along with their review(positive ,negative, neutral).	Pass

8.9 Weka Tool Testing:

8.9.1 Classification:

Classification consists of predicting a certain outcome based on a given input. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and the respective outcome, usually called goal or prediction attribute. The algorithm tries to discover relationships between the attributes that would make it possible to predict the outcome. Next the algorithm is given a dataset not seen before, called prediction set, which contains the same set of attributes, except for the prediction attribute–not yet known. The algorithm analyses the input and produces a prediction. The prediction accuracy defines how “good” the algorithm is.

NAIVE BAYES

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 3" in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the colour, roundness and diameter features.

Abstractly, Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some *n* features
(dependent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

For each of *k* possible outcomes or *classes*.

The problem with the above formulation is that if the number of features *n* is large or

If a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k | \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} | C_k)}{p(\mathbf{x})}.$$

LINEAR REGRESSION

In statistics, linear regression is an approach for modelling the relationship between a scalar dependent variable *y* and one or more explanatory variables (or independent variable) denoted *X*. The case of one explanatory variable is called *simple linear regression*.
In linear regression, data are modelled using linear predictor functions, and unknown model parameters are estimated from the data. Such models are called *linear models*

Given a data set $\{y_i,\ x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of *n* statistical units, a linear regression model assumes that the relationship between the dependent variable y_i and the *p*-vector of regressor’s x_i is linear. This relationship is modelled through a *disturbance term* or *error variable* ε_i — an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol\beta + \varepsilon_i, \qquad i = 1, \ldots, n,$$

Where $^{\mathrm{T}}$ denotes the transpose, so that $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol\beta$ is the inner product between vectors $\boldsymbol{x}_i$ and $\boldsymbol\beta$. Often these *n* equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol\beta + \boldsymbol\varepsilon,$$

Where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

y_i Is called the *regress and*, *endogenous variable*, *response variable*, *measured variable*, *criterion variable*, or *dependent variable*.

$x_{i1}, x_{i2}, \ldots, x_{ip}$ Are called *regressors*, *exogenous variables*, *explanatory variables*, *covariates*, *input variables*, *predictor variables*, or *independent variables*

β Is a *p*-dimensional *parameter vector*. Its elements are also called *effects*, or *regression coefficients*.

ε_i Is called the *error term*, *disturbance term*, or *noise*.

8.9.2 Clustering:

Clustering can be considered the most important unsupervised learning problem; so, as every other problem of this kind, it deals with finding a structure in a collection of unlabelled data. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

SIMPLEK-MEANS

This algorithm randomly selects K number of objects, each of which initially represents a cluster mean or centre. For each of the remaining objects, an object is assigned to the cluster to which it is most similar, based on the distance between the object and cluster mean. Then it computes new mean for each cluster. This process iterates until the criterion function converges.

Given an initial set of *k* means $m_1^{(1)},\ldots,m_k^{(1)}$ (see below), the algorithm proceeds by alternating between two steps:[7]

Assignment step: Assign each observation to the cluster whose mean yields the least

Within-cluster sum of squares (WCSS). Since the sum of squares is the squared Euclidean distance, this is intuitively the "nearest" mean.[8] (Mathematically, this means partitioning the observations according to the Voronoi diagram generated by the means).

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\|^2 \le \|x_p - m_j^{(t)}\|^2 \ \forall j, 1 \le j \le k\},$$

Where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them.

Update step: Calculate the new means to be the centroids of the observations in the new clusters.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

HIERARCHICAL CLUSTERING

In data mining, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- **Agglomerative**: This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- **Divisive**: This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In the general case, the complexity of agglomerative clustering is $O(n^3)$, which makes them too slow for large data sets. Divisive clustering with an exhaustive search is $O(2^n)$, which is even worse.

Classification Testing Results Table

Algorithm Name	**Time taken (in sec)**	**Results on dataset**
Naive Bayes	0.04	Correctly Classified Instances =100% Incorrectly Classified Instances = 0% Mean absolute error = 0 Root mean squared error = 0
Linear Regression	0.07	Mean absolute error =37.0178 Root mean squared error =67.04 Relative absolute error =100 % Root relative squared error =100 %

Related terms:

Mean absolute error: Mean absolute error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes.

Root-mean-square error (RMSE) : is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed.

In case of Naïve Bayes algorithm the above mentioned error rates are zero but in case of Linear Regression they are 37.01 and 67.04 respectively. So Naive Bayes is error free.

Based on the accuracy level of 100% in Naïve Bayes, this is a pretty good model to predict whether land owner can get a profit or not.

Clustering Testing Results Table

Algorithm Name	**Time taken (in sec)**	**Clustered Instances**	**Results on dataset**
SimpleK-Means	0.01	0 - 22 (76%) 1 - 7 (24%)	Number of iterations = 7 Within cluster sum of squared errors = 32.962504527908564
Hierarchical Clusterer	0.02	0 - 28 (97%) 1 - 1 (3%)	NULL

In case of SimpleK-means algorithm in clustering we end up with the error of 32.96, but Naïve Bayes algorithm looks better with 0% error. When this comparison we conclude that Naïve Bayes is the best algorithm for our prediction among the other four analysed algorithms.

CONCLUSION

The most commonly used keyword extraction algorithm when a document is given is TFIDF. Along with this the Text rank algorithm will be supportive and helpful to depict the graph of actually what and how the extraction is going on. As early mentioned that the methods in extraction the words are the supportive tool for the text summarization and will be very helpful in better understanding of the document. Here nearly all algorithms for word extraction depend on a weighed function which actually balances some measures of term within the document. Overall, many examples with this collection of approaches gave us the overall picture of the challenges of the analysis on the project and was helpful in getting the sentiment of the document. First we note the use of classification and by implementing the algorithms we faced many challenges in extracting keywords such as eliminating the foreign keys, ranking the text blocks etc.

FUTURE ENHANCEMENT

- An option will be given to upload a file (.pdf .word .txt formats).
- An option will be given to upload a photo where the app will scan the photo and extract the content from that and gives us the summary and the extracted words.

REFERENCES

1: https://www.microsoft.com/en-us/research/publication/automatic-extraction-of-titles-fromgeneral-documents-using-machine-learning/

2: https://www.researchgate.net/publication/222554288_Automatic_extraction_of_titles_from_g eneral_documents_using_machine_learning

3: https://www.burakkanber.com/blog/machine-learning-sentiment-analysis/

4: http://cs229.stanford.edu/proj2010/HsuSeeWu-MachineLearningForSentimentAnalysis.pdf

5: Koppel, Moshe; Schler, Jonathan (2006). "The Importance of Neutral Examples for Learning Sentiment". Computational Intelligence.

6: Cambria, E; Schuller, B; Xia, Y; Havasi, C (2013). "New avenues in opinion mining and sentiment analysis. IEEE Intelligent Systems.

7: Liu, Bing (2010). "Sentiment Analysis and Subjectivity In Indurkhya, N.; Damerau, F. J. Handbook of Natural Language Processing (Second ed.).