University of Dhaka

Institute of Information Technology

Program: Master in Information Technology

Course: Advanced Object Oriented Programming

Project: Text Summarizer

Submitted By

Md. Siam Ansary	Roll: 201103	Email: siamansary.cse@gmail.com
Kanok Chanpa Saha Bhowmik	Roll: 201120	Email: rinky.saha360@gmail.com
Atonu Saha	Roll: 201127	Email: atonuewu@gmail.com
Umme Kawser Sinthia	Roll: 201136	Email: sinthy08@gmail.com

TABLE OF CONTENTS

TABLE OF CONTENTS i
LIST OF FIGURES ii

1 Introduction 01
1.1 Text Summarization 01
1.2 Object Oriented Programming 01
1.3 "Text Summarizer" Application 01

2 Automatic Text Summarization Implementation Approach 02

3 Project Implementation 03
3.1 Implementation Aspects 03
3.1.1 User registration and log in 02
3.1.2 Input for summarization 03
3.1.3 Summary Generation 03
3.1.4 Generated summary storing 04
3.2 Implementation Packages and Classes 05

4 Database 10

5 Limitations 10
5.1 Limited input formats 10
5.2 Limited criteria for summarization 10
5.3 Limited accuracy of generated summaries 11
5.4 Local database 11

6 Samples of summarization 11
6.1 First Sample 11
6.2 Second Sample 12

7 Source Code 13

8 References 13

LIST OF FIGURES

Figure 3.1.3.1	Use of multi-threading for summary generation	04
Figure 3.2.1	Package Diagram of implementation	05
Figure 3.2.2	Class Diagram of implementation	09

1. Introduction

1.1.Text Summarization

Every single day enormous amount of data is growing and much of it is in textual form. There is great need to reduce much of this text data to shorter, focused summaries that capture salient details so that they can be utilized properly. With the progress of artificial intelligence and natural language processing, the topic of text summarization has gained much attention and enthusiasm. Developing a text summarizer that produces result close to human generated summary is of huge significance.

A summary is an overview of content that provides a reader with overarching theme but does not expand on specific details. Text summarization is a way to condense the large amount of information into a concise form by the process of selection of important information and discarding unimportant and redundant information. Automatic text summarization is the process of summarization without human involvement and it is mainly of two kinds. They are extractive approach and abstractive approach. The extractive summarization is the one where the exact sentences present in the document are used as summaries whereas the abstractive summarization is the process in which the abstract of the document is created.

1.2.Object Oriented Programming

Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behavior.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. This approach to programming is well-suited for programs that are large, complex and actively updated or maintained.

1.3.“Text Summarizer” Application

Here, an application “Text Summarizer” has been implemented using Java programming language through the Object Oriented Programming approach. This implementation works as an extractive summarizer and works with sentiment analysis and sentence length.

2. Automatic Text Summarization Implementation Approach

In the project work, Java programming language has been used to do extractive summarization of a single document input.

Extractive text summarization is an ongoing research topic of Natural Language Processing field. Many researchers have tried different approaches for better summary generation process. Here inspirations from different researches have been taken.

Dabholkar, Patadia and Dsilva [1] discussed about extractive document summarization which is done with sentiment analysis. Subjective or objective information is usually differentiated on basis of presence of certain elements. Text element with neutral sentimental tone may be categorized as objective. They emphasized that neutral sentences don’t have any positive or negative opinions or biased information, rather they hold facts about a matter. Hence, identifying neutral sentences, they should be included in the summary. Solov’ev, Antonova and Pazel’skaia [2] talked about text summarization that is also sentiment based. They prioritized fragments with highest emotional charge. Sometimes sentiment bearing sentence may appear to be neutral if weakly expressed. This approach selects non-neutral sentences for the summarization purpose using absolute value of sentiment scores. Srivastava and Gupta [3] discussed a text summarization approach where it is said that summaries should be adjusted according to sentence lengths as compression ratio should not be compromised.

Emphasis has been given to sentiment analysis and sentence length when implementing. Through sentiment analysis, sentences have been categorized as neutral, positive, negative, very positive and very negative. This has been done using Stanford NLP [4] libraries. Stanford NLP involves training a complicated recursive neural network. The neural network is trained on a sentiment tree-bank. This teaches the neural network to associate subparts of sentences with sentiment labels incrementally, combining the sentiment labels of smaller phrases into sentiment labels for larger phrases that combine them, rather than trying to predict a sentiment label for an entire sentence at once. This Stanford CoreNLP's approach does not support sentiment analysis of chunks containing multiple sentences. Neutral, very positive and very negative sentences have been considered as summary sentences. Also, the sentences whose lengths are not very long in a document have been considered as summary sentences. Very short sentences may not contain important data and very long sentences can compromise data compression ratio. The sentences meeting both condition have been used to form summaries. Multi-threading has been used to do select candidate summary sentences from sentiment analysis and sentence length analysis.

3. Project Implementation

3.1.Implementation Aspects

3.1.1. User registration and log in

To use the application one has to be a registered user. To register, one needs to input first name, surname, username and password. The username and the password need to be of certain length. Also, more than one user cannot have the same username. Using the username and password, one can log into the app.

3.1.2. Input for summarization

A user can input a text in two ways. The user can browse a text file or copy and paste the input into text field. The input needs to be of certain length to avoid erroneous input and against each input, user must put a subject for the summary input. From the subject, an identification number will be generated by combining the subject and the date-time of summary generation.

3.1.3. Summary generation

After the input is complete, to identify candidate summary sentences, two more threads are created. One of the threads identify candidate sentences with sentiment analysis and another thread identify with sentence length analysis. Then summary is created with the sentences meeting all conditions. So, the input is taken in main thread, then, two other created threads work on identifying candidate summary sentences. After that, main thread forms the summary from the candidate sentences. After summary is generated, the input and the summary are displayed with other information and a folder is created on the Desktop of the user's computer inside which in a PDF file, user's name, input text and the summary are saved.

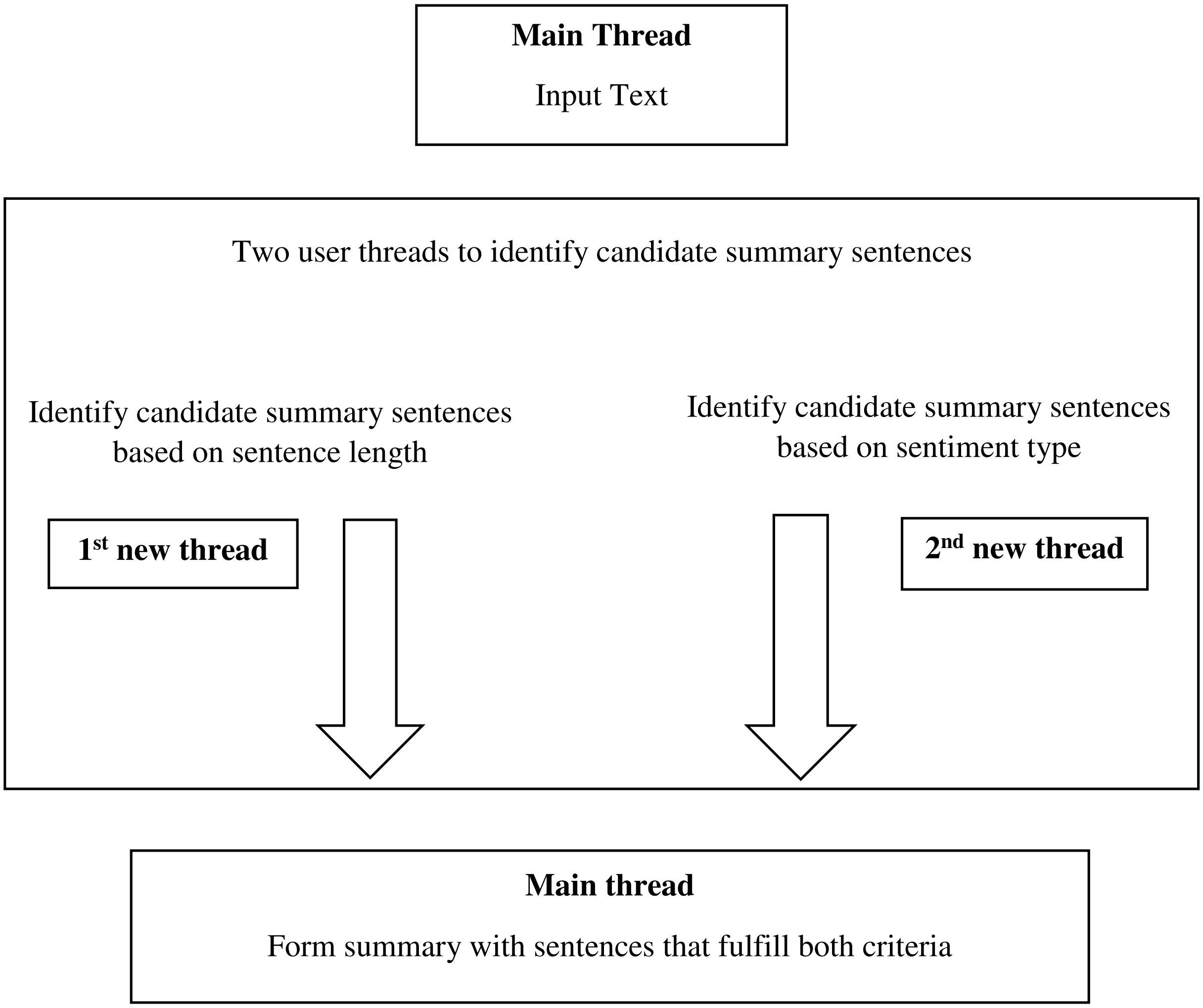
Main Thread
Input Text
Two user threads to identify candidate summary sentences
Identify candidate summary sentences based on sentence length
Identify candidate summary sentences based on sentiment type
1st new thread
2nd new thread
Main thread
Form summary with sentences that fulfill both criteria

Figure 3.1.3.1: Use of multi-threading for summary generation

3.1.4. Generated summary storing

After a summary is generated, a folder will be created on the Desktop of the personal computer of the user inside which in a PDF file, user's name, input and summary can be found. Also, the user has the option to store the summary in the database of the application. A user can find the previously generated summaries' list and using the identification number can check the input, summary with date-time and convert the record into a PDF.

3.2.Implementation Packages and Classes

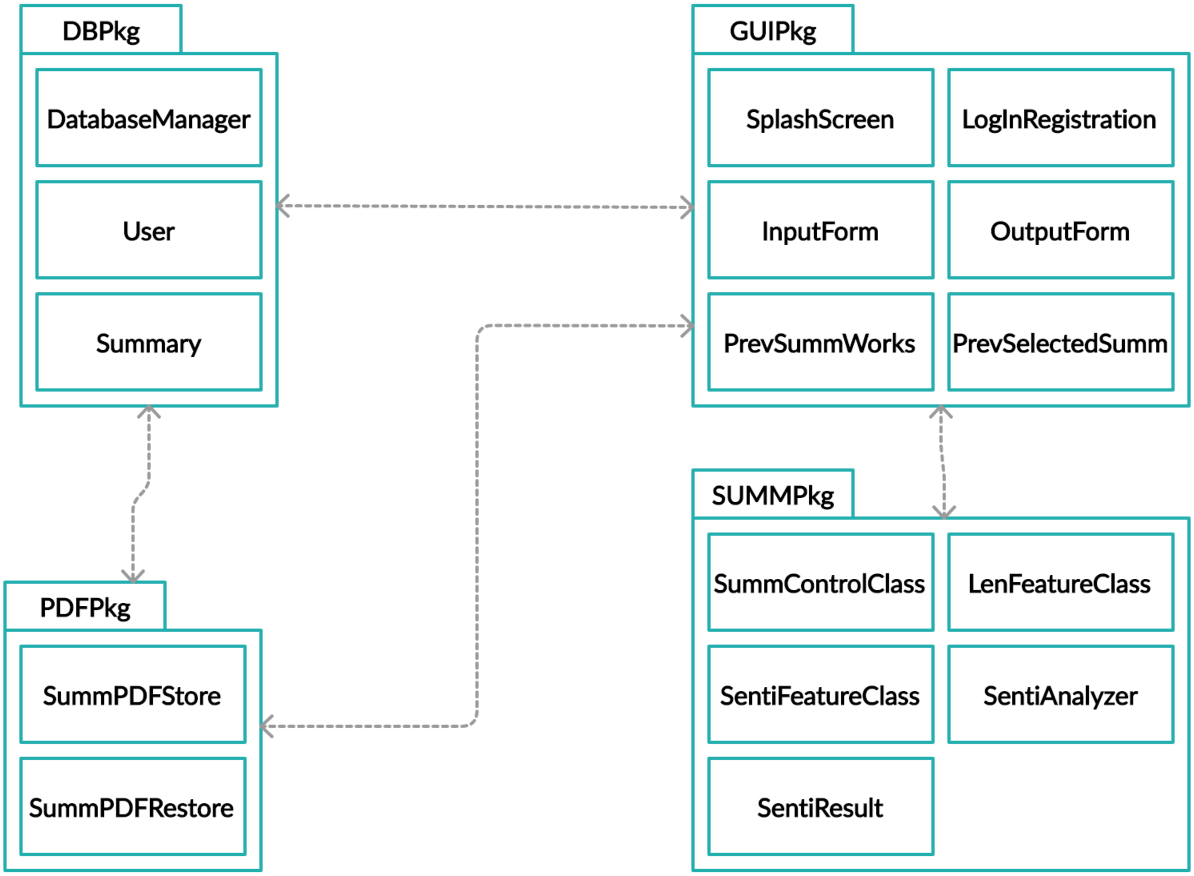

Figure 3.2.1: Package Diagram of implementation *

In the implementation, the coding has been done through four packages. Each package has several classes. The packages and their classes are described as follows.

a) **GUIPkg:** The classes of this package are mainly related to the Graphical User Interface of the project. All the classes extend JFrame class. There are six classes under this package. They are briefly described in the following.

* Designed using Creately, www.creately.com

i. **SplashScreen:** When the project is run, a user is welcomed through the frame of SplashScreen class. It then leads the user to the frame of LogInRegistration class.

ii. **LogInRegistration:** If a new user is using who is not registered, then, s/he has to register using first name, surname, username and password. Username and password need to be at least of length six. When a user tries to register, one has to use a username that is not already in use. If the username is already in use, user will be informed. This is done by checking the database of the project that holds information of all registered users. To log in, one has to provide username and password. If username and password don't match or if they are not in database, user will be notified. After successful log in, user will go to the frame of InputForm class through which new summarization can be done or previous works can be accessed.

iii. **InputForm:** In the frame of InputForm class, user can do several things. If user wants to summarize, s/he can provide an input. This can be done through two ways. A text file can be chosen or input can be given directly. The input text should be at least of six hundred characters length. Also, the user has to give a subject name for the summarization work which should be at least of length four. Subject is needed because it is used to identify summaries when a user wants to access previous works. The identification for a summary is created from the subject and the date-time of the summarization work. After summarization is done and recorded in PDF file with necessary information, user is sent to the frame of OutputForm class. If a user wants to access previous works, by a button click can go to the frame of PrevSummWorks class which lists the previous summaries done by the user.

iv. **OutputForm:** After the summary is generated and recorded in PDF format with important information, user is sent to the frame of OutputForm class. User can see input text and generated summary along with other data. S/he can open the generated PDF file of summarization by a button click and also if wants, can insert the summary with other information into the database.

v. **PrevSummWorks:** For the logged in registered user, the frame of PrevSummWorks class shows the list of subjects, dates and identifications of the previously stored generated summaries. User can access the input text and summary by inputting the identification from the list. The inputted identification is searched in database. If a proper identification is given, user can access the previous work, otherwise will be notified properly.

vi. **PrevSelectedSumm:** If the user inputs a proper identification, s/he can see the input and summary in the frame of PrevSelectedSumm class. There is also option to save the summary with other data in PDF format.

b) **SUMMPkg:** The classes of this package are mainly related to generation of summary using length feature and sentiment analysis. There are five classes in this package. They are described below in brief manner.

i. **SummControlClass:** SummControlClass class is executed in main thread. It creates objects of LenFeatureClass class and SentiFeatureClass class who extend the Thread class. The input text is passed to them as argument and the main thread gives the user thread opportunities to work. After LenFeatureClass and SentiFeatureClass class identifies candidate summary sentences return control to main thread, SummControlClass class forms summary with the sentences that fulfill both aspects. Then the summary is passed back to InputForm class.

ii. **LenFeatureClass:** This class extends the Thread class. It calculates the lengths of the sentences and average length of the sentences of the input text. Then, identifies the sentences that are slightly longer than average length as candidate sentences. This works as a user thread along with SentiFeatureClass class.

iii. **SentiFeatureClass:** SentiFeatureClass class extends the Thread class. It sends the input text to SentiAnalyzer class that categories sentences into neutral, very positive, positive, very negative and negative and sets the categories in SentiResult. Using the sentences that have been identified as neutral, very positive and very negative, this class selects candidate summary sentences.

iv. **SentiAnalyzer:** SentiAnalyzer class uses StanfordCoreNLP to determine the sentiment types of each sentence of the input text. By splitting the sentence, tokenization, creating a parse tree of the sentence by means of a complicated recursive neural network sentiment type is determined and set with SentiResult class object.

v. **SentiResult:** The sentiment type of any string of the input text is set and can be checked with the help of this class.

c) **PDFPkg:** The classes of this package are used to save an input and its summary with other information in a PDF file in a particular folder. There are two classes in this package. They are described below in brief manner.

i. **SummPDFStore:** After summary is generated, InputForm class object is passed to this class and a PDF file is generated that has the user's name, input text and the summary. The PDF is saved in a folder that is named after the date-time of the operation. The location of the PDF is then passed to InputForm class.

ii. **SummPDFRestore:** If the user want to save a previous work in PDF, this class is used. It saves the user's name, input text and summary in PDF with proper formatting. User can access the PDF with just a button click.

d) **DBPkg:** The classes of this package are mainly related to storing an input and its summary with other information in the database. There are three classes in this package. There are described below in concise manner.

i. **DatabaseManager:** Database operations are done in this class. This project's database has two tables. One table holds information about users and another holds information about generated summaries. With the methods of this class, users or summaries can be added, search and retrieved.

ii. **Summary:** There are input text, generated summary, username, subject, date and identification for a summary operation. With the Summary class they can be set and retrieved.

iii. **User:** For every registered user, there are first name, surname, password and username. These are set and can be retrieved with the help of this User class.

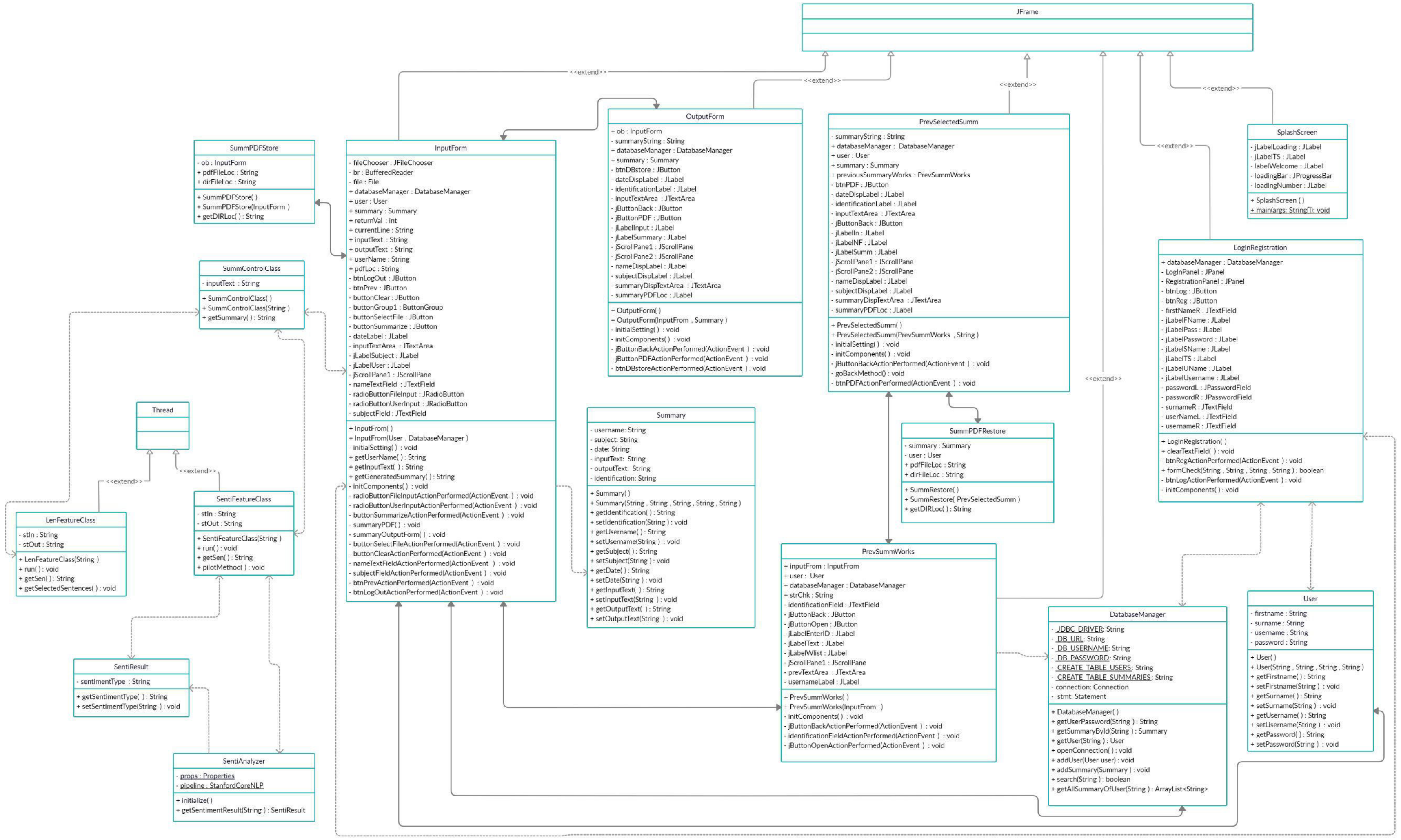
JFrame
<<extend>>
<<extend>>
<<extend>>
<<extend>>
<<extend>>
<<extend>>
SummPDFStore
- ob : InputForm
+ pdfFileLoc : String
+ dirFileLoc : String
+ SummPDFStore( )
+ SummPDFStore(InputForm)
+ getDIRLoc( ) : String
InputForm
- fileChooser : JFileChooser
- br : BufferedReader
- file : File
+ databaseManager : DatabaseManager
+ user : User
+ summary : Summary
+ returnVal : int
+ currentLine : String
+ inputText : String
+ outputText : String
+ userName : String
+ pdfLoc : String
- btnLogOut : JButton
- btnPrev : JButton
- buttonClear : JButton
- buttonGroup1 : ButtonGroup
- buttonSelectFile : JButton
- buttonSummarize : JButton
- dateLabel : JLabel
- inputTextArea : JTextArea
- jLabelSubject : JLabel
- jLabelUser : JLabel
- jScrollPane1 : JScrollPane
- nameTextField : JTextField
- radioButtonFileInput : JRadioButton
- radioButtonUserInput : JRadioButton
- subjectField : JTextField
+ InputFrom( )
+ InputFrom(User , DatabaseManager)
- initialSetting( ) : void
+ getUserName( ) : String
+ getInputText( ) : String
+ getGeneratedSummary( ) : String
- initComponents( ) : void
- radioButtonFileInputActionPerformed(ActionEvent) : void
- radioButtonUserInputActionPerformed(ActionEvent) : void
- buttonSummarizeActionPerformed(ActionEvent) : void
- summaryPDF( ) : void
- summaryOutputForm( ) : void
- buttonSelectFileActionPerformed(ActionEvent) : void
- buttonClearActionPerformed(ActionEvent) : void
- nameTextFieldActionPerformed(ActionEvent) : void
- subjectFieldActionPerformed(ActionEvent) : void
- btnPrevActionPerformed(ActionEvent) : void
- btnLogOutActionPerformed(ActionEvent) : void
OutputForm
+ ob : InputForm
- summaryString : String
+ databaseManager : DatabaseManager
+ summary : Summary
- btnDBstore : JButton
- dateDispLabel : JLabel
- identificationLabel : JLabel
- inputTextArea : JTextArea
- jButtonBack : JButton
- jButtonPDF : JButton
- jLabelInput : JLabel
- jLabelSummary : JLabel
- jScrollPane1 : JScrollPane
- jScrollPane2 : JScrollPane
- nameDispLabel : JLabel
- subjectDispLabel : JLabel
- summaryDispTextArea : JTextArea
- summaryPDFLoc : JLabel
+ OutputForm( )
+ OutputForm(InputFrom , Summary)
- initialSetting( ) : void
- initComponents( ) : void
- jButtonBackActionPerformed(ActionEvent) : void
- jButtonPDFActionPerformed(ActionEvent) : void
- btnDBstoreActionPerformed(ActionEvent) : void
PrevSelectedSumm
- summaryString : String
+ databaseManager : DatabaseManager
+ user : User
+ summary : Summary
+ previousSummaryWorks : PrevSummWorks
- btnPDF : JButton
- dateDispLabel : JLabel
- identificationLabel : JLabel
- inputTextArea : JTextArea
- jButtonBack : JButton
- jLabelIn : JLabel
- jLabelNF : JLabel
- jLabelSumm : JLabel
- jScrollPane1 : JScrollPane
- jScrollPane2 : JScrollPane
- nameDispLabel : JLabel
- subjectDispLabel : JLabel
- summaryDispTextArea : JTextArea
- summaryPDFLoc : JLabel
+ PrevSelectedSumm( )
+ PrevSelectedSumm(PrevSummWorks , String)
- initialSetting( ) : void
- initComponents( ) : void
- jButtonBackActionPerformed(ActionEvent) : void
- goBackMethod() : void
- btnPDFActionPerformed(ActionEvent) : void
SplashScreen
- jLabelLoading : JLabel
- jLabelTS : JLabel
- labelWelcome : JLabel
- loadingBar : JProgressBar
- loadingNumber : JLabel
+ SplashScreen ( )
+ main(args: String[]): void
LogInRegistration
+ databaseManager : DatabaseManager
- LogInPanel : JPanel
- RegistrationPanel : JPanel
- btnLog : JButton
- btnReg : JButton
- firstNameR : JTextField
- jLabelFName : JLabel
- jLabelPass : JLabel
- jLabelPassword : JLabel
- jLabelSName : JLabel
- jLabelTS : JLabel
- jLabelUName : JLabel
- jLabelUsername : JLabel
- passwordL : JPasswordField
- passwordR : JPasswordField
- surnameR : JTextField
- userNameL : JTextField
- usernameR : JTextField
+ LogInRegistration( )
+ clearTextField( ) : void
- btnRegActionPerformed(ActionEvent) : void
+ formCheck(String , String , String , String) : boolean
- btnLogActionPerformed(ActionEvent) : void
- initComponents( ) : void
SummControlClass
- inputText : String
+ SummControlClass( )
+ SummControlClass(String)
+ getSummary( ) : String
Summary
- username: String
- subject: String
- date: String
- inputText: String
- outputText: String
- identification: String
+ Summary( )
+ Summary(String , String , String , String , String)
+ getIdentification( ) : String
+ setIdentification(String) : void
+ getUsername( ) : String
+ setUsername(String) : void
+ getSubject( ) : String
+ setSubject(String) : void
+ getDate( ) : String
+ setDate(String) : void
+ getInputText( ) : String
+ setInputText(String) : void
+ getOutputText( ) : String
+ setOutputText(String) : void
SummPDFRestore
- summary : Summary
- user : User
+ pdfFileLoc : String
+ dirFileLoc : String
+ SummRestore( )
+ SummRestore(PrevSelectedSumm)
+ getDIRLoc( ) : String
<<extend>>
Thread
<<extend>>
<<extend>>
LenFeatureClass
- stIn : String
- stOut : String
+ LenFeatureClass(String)
+ run( ) : void
+ getSen( ) : String
+ getSelectedSentences( ) : void
SentiFeatureClass
- stIn : String
- stOut : String
+ SentiFeatureClass(String)
+ run( ) : void
+ getSen( ) : String
+ pilotMethod( ) : void
PrevSummWorks
+ inputFrom : InputFrom
+ user : User
+ databaseManager : DatabaseManager
+ strChk : String
- identificationField : JTextField
- jButtonBack : JButton
- jButtonOpen : JButton
- jLabelEnterID : JLabel
- jLabelText : JLabel
- jLabelWlist : JLabel
- jScrollPane1 : JScrollPane
- prevTextArea : JTextArea
- usernameLabel : JLabel
+ PrevSummWorks( )
+ PrevSummWorks(InputFrom)
- initComponents( ) : void
- jButtonBackActionPerformed(ActionEvent) : void
- identificationFieldActionPerformed(ActionEvent) : void
- jButtonOpenActionPerformed(ActionEvent) : void
DatabaseManager
- JDBC_DRIVER: String
- DB_URL: String
- DB_USERNAME: String
- DB_PASSWORD: String
- CREATE_TABLE_USERS: String
- CREATE_TABLE_SUMMARIES: String
- connection: Connection
- stmt: Statement
+ DatabaseManager( )
+ getUserPassword(String) : String
+ getSummaryById(String) : Summary
+ getUser(String) : User
+ openConnection( ) : void
+ addUser(User user) : void
+ addSummary(Summary) : void
+ search(String) : boolean
+ getAllSummaryOfUser(String) : ArrayList<String>
User
- firstname : String
- surname : String
- username : String
- password : String
+ User( )
+ User(String , String , String , String)
+ getFirstname( ) : String
+ setFirstname(String) : void
+ getSurname( ) : String
+ setSurname(String) : void
+ getUsername( ) : String
+ setUsername(String) : void
+ getPassword( ) : String
+ setPassword(String) : void
SentiResult
- sentimentType : String
+ getSentimentType( ) : String
+ setSentimentType(String) : void
SentiAnalyzer
- props : Properties
- pipeline : StanfordCoreNLP
+ initialize( )
+ getSentimentResult(String) : SentiResult

Figure 3.2.2 : Class Diagram of implementation *

* Designed using Creately, "www.creately.com"

4. Database

The database used in implementation is H2. It is a free SQL database written in Java. It can be easily embedded in Java applications. Two tables have been created in the database to store information. The tables are USERS and SUMMARIES.

SUMMARIES (summaryid, username, subject, identification, date, input, output)
USERS (userid, firstname, surname, username, password)

In the USERS table, information about different registered users are kept. When a new user wants to register, first name, surname, username and password have to be provided of certain lengths. If the username is already in database, user will be notified after checking the database as multiple user cannot have same username and if registration is successful, information of the user is inserted into database. While logging, provided username and password are checked in database and if both are matched, user will be proceeded forward. In the SUMMARIES table, generated summaries are kept along with username, subject of the summary, identification, date and input text. When generating a summary, user has the option to store the summary in the database so that that summary can be retrieved afterwards and be converted into PDF.

5. Limitations

5.1.Limited Input Formats

In the work, pre-processing phase of an input has not been implemented. As a result, not all texts can be worked with. Some texts can lead to failures when storing a summary into database. Through browsing, user can select a text file as an input but other file formats cannot be worked with. Also, this is a single document summarization and only can deal with simple formats. As pre-processing phase has not been implemented, pre-processed inputs need to be used to create a summary.

5.2.Limited Criteria for Summarization

In current times, many criteria are being considered to create summaries. Many ranking algorithms, artificial intelligence approaches and hybrid approaches are being experimented with. This implementation has not worked with many important criteria.

5.3.Limited Accuracy of Generated Summaries

The accuracy for the generated summaries of the implementation has not been checked using any dataset. As this implementation has not worked with many research criteria, when checked, accuracy might not meet expectations.

5.4.Local Database

The generated summaries can be stored using a local database system. As cloud databases are gaining more popularity and machines can break down unexpectedly, a process to store summaries using online databases might be a good additional option.

6. Samples of Summarization

6.1.First Sample

Input Text: Japanese culture has its own unique forms of comic books and animation. Manga and anime are extremely popular in Japan. The earliest animation that is known to have been created in Japan was released in 1917. This early cartoon featured a samurai testing a sword and being defeated. The modern style of anime was developed during the 1960s. One of the most influential artists is Osamu Tezuka.

All genres are represented, but science fiction is by far the most popular. Manga features similar content. Tezuka continued to shape the manga and anime industries over the years. Many of the common characters, like giant robots, come from his influence. Giant robots were further developed by Go Nagai and other animators into a new genre called Super Robot. This genre evolved through the work of Yoshiyuki Tomino and became known as Real Robot. The 1980s brought many classic animes in this genre, like The Super Dimension Fortress Macross and Gundam films. Anime obtained vast mainstream acceptance throughout Japan in the 1980s.

People who are familiar with American comic books will know that women are a small minority of its reading audience. Popularity with women has helped manga spread rapidly outside of Japan. Shojo can also include exciting action stories with strong female protagonists in interesting roles. A common characteristic of manga are females with huge expressive eyes. Supernatural elements are also popular in this reading material. Offbeat subjects are another theme that is common in some shoji manga.

Generated Summary: The earliest animation that is known to have been created in Japan was released in 1917. This early cartoon featured a samurai testing a sword and being defeated. Manga features similar content. Tezuka continued to shape the manga and anime industries over the years. This genre evolved through the work of Yoshiyuki Tomino and became known as Real Robot. Popularity with women has helped manga spread rapidly outside of Japan.

6.2.Second Sample

Input Text: Lebanon was already suffering a major economic downturn before the explosion which left at least 154 people dead with 5000 injured and 300000 homeless.

The World Food Programme said the damage to Beirut port would interrupt food supplies and push prices up. The World Health Organization said the health system was seriously damaged with three hospitals out of action. Meanwhile the Lebanese President Michel Aoun rejected calls for an international investigation into the explosion and said local authorities would examine whether it was triggered by external interference such as a bomb. The leader of the militant Hezbollah movement also denied allegations that it had stored weapons or ammunition at the port.

The government has said the blast was the result of the detonation of 2750 tonnes of ammonium nitrate that had been stored unsafely at the port for six years.The decision to keep so much explosive material in a warehouse near the city centre has been met with disbelief and fury by many Lebanese who have long accused the political elite of corruption and mismanagement.

WFP spokeswoman Elisabeth Byrs told reporters in Geneva that the organisation was concerned the severe damage to Beirut port could limit the flow of food supplies and push prices beyond the reach of many. The WFP was sending 5000 food parcels that would be enough to feed a family of five for a month and was planning to import wheat flour and grains. Christian Lindmeier of the WHO meanwhile warned that the hospitals were overwhelmed with the patients.

The WHO has delivered emergency trauma and surgical supply kits containing essential medicines and medical supplies and is calling for financial support to cover immediate needs and ensure continuity in the response to the Covid-19 pandemic. Many countries have offered aid to help Lebanon with the US announcing on Friday that it planned to immediately send food and medicine.

Lebanese president and prime minister have said the ammonium nitrate which is commonly used as a fertiliser but can also made into an explosive had been stored in a warehouse at the port without any safety precautions since 2014 when it was unloaded from an impounded cargo ship. Officials have said the explosion appears to have been triggered by a fire and there has been no evidence so far of the third possibility.

Generated Summary: The World Health Organization said the health system was seriously damaged with three hospitals out of action. Many countries have offered aid to help Lebanon with the US announcing on Friday that it planned to immediately send food and medicine.

7. Source Code

The source code of the project can be found at the link https://tinyurl.com/textsummarizer

8. References

1. Salil Dabholkar, Yuvraj Patadia and Prajyoti Dsilva, “Automatic Document Summarization using Sentiment Analysis”, in Proceedings of 2016 International Conference on Informatics and Analytics (ICIA), August, 2016, Pondicherry, India.

2. Solov’ev A.N., Antonova A.Ju. and Pazel’skaia A.G., “Using Sentiment Analysis for Text Information Extraction”, in proceedings of the 2012 Computational Linguistics and Intelligent Technology, pp. 616-627, August, 2012, Moscow, Russia.

3. Noopur Srivastava and Bineet Kumar Gupta, “An Algorithm for Summarization of Paragraph Upto One Third with the Help of Cue Words Comparison”, International Journal of Computer Science and Applications, Volume 5, No. 5, pp. 167-171, 2014, United Kingdom.

4. “The Stanford Natural Language Processing Group”, https://nlp.stanford.edu/ , Last accessed August 9, 2020