VOICE-BASED DOCUMENT SEARCH AND INTERPRETATION FOR VISUALLY IMPAIRED

A PROJECT REPORT

Submitted by

PRITISH RAWAL [Reg No: RA141100- 8010085]
ATULESH PANDEY [Reg No: RA1411008010093]
TRIDHA CHAUDHURI [Reg No: RA1411008010126]

Under the guidance of

Dr. G. VADIVU, Ph.D

(Professor & Head , Department of INFORMATION TECHNOLOGY)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Kancheepuram District

APRIL 2018

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**VOICE-BASED DOCUMENT SEARCH AND INTERPRETATION FOR VISUALLY IMPAIRED**" is the bonafide work of " **PRITISH RAWAL [Reg No: RA141100-8010085], ATULESH PANDEY [Reg No: RA1411008010093], TRIDHA CHAUDHURI [Reg No: RA1411008010126]**", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. G. VADIVU, Ph.D
GUIDE
Professor & Head
Dept. of INFORMATION TECHNOLOGY

SIGNATURE

Dr. G. VADIVU
HEAD OF THE DEPARTMENT
Dept. of Information Technology

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

A significant percentage of the human population is visually impaired and faces multiple problems in day to day life. For them, reading documents, forms and pictures becomes demanding. This system helps visually impaired people and people with reading disorders by converting the needed text into speech and reads it out for the user. Moreover, the user is allowed to choose from multiple options such as summarization of the text and finding the topics available in the document with the use of LDA. The input given by the user can be either by text or by speech, helping those who are visually impaired to interact with the system easily. Lastly, for more information on any of the topics, a Google search and a web browser history search is done. The proposed application integrates topic modeling, text to speech and speech recognition in order to provide an interactive environment which makes visually impaired people and people with reading disorders to be more self dependent.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Dr. G. Vadivu for her valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of her busy schedule, she has extended cheerful and cordial support to me for completing this research work.

Author

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

ABBREVIATIONS x

1 INTRODUCTION 1

2 LITERATURE SURVEY 2

2.1 Overview of Tesseract 2

2.2 Processing 2

2.3 Wearable Face Recognition System For Visually Impaired People . 3

2.4 Advanced GPS and GSM Navigation for Blinds 3

3 System Analysis 5

4 System Design 6

5 Methodology 8

5.1 Image Processing 8

5.1.1 Gray Scale Conversion 9

5.1.2 Thresholding 9

5.2 Image to Text 11

5.2.1 Tesseract 11

5.2.2 Pytesseract 12

5.3 Autocorrection 13

5.3.1 Autocorrect 14
5.4 Text Summarization 14
5.4.1 Initializing The Range And Stop Words 15
5.4.2 Computing Frequency of Each Word 15
5.4.3 Generating The Summary 15
5.4.4 Printing The Summary 16
5.5 Topic Modelling 16
5.5.1 Parameters of LDA 17
5.5.2 Cleaning and Processing 18
5.5.3 Preparing Document Term Matrix 18
5.5.4 Running The LDA Model 18
5.6 Speech To Text 18
5.6.1 Speechrecognition 19
5.7 Text To Speech 20
5.7.1 Windows Search Engine 20
5.7.2 Win32com 21
5.8 Accessing Internet And Web Browsing History 21
5.8.1 Accessing Internet 21
5.8.2 Searching Web Browser History 21

6 Coding and Testing **23**
6.1 Coding 23
6.2 Testing 36

7 Result **39**
7.1 Speech To Text 39
7.2 Extracting Text From Image 39
7.3 Autocorrect 41
7.4 Summarization 41
7.5 Topic Modelling 42
7.6 Text to Speech 42
7.7 Searching Related Document 43

8 Future Enhancement **44**

9 Conclusion **45**

LIST OF TABLES

6.1 Table for Test Cases (continued on the next page) 36
6.2 Table for Test Cases (continued on the next page) 37
6.3 Table for Test Cases 38

LIST OF FIGURES

4.1 Flowchart of system design 7

5.1 RGB to grayscale conversion 9
5.2 OTSU's Binarization 10
5.3 OTSU's binarization equation 10
5.4 Erosion 11
5.5 Steps of working of tesseract 12
5.6 The image on the left is converted to text using pytesseract in python 13
5.7 Autocorrection 14
5.8 Document term matrix 16
5.9 Topic term matrix 17
5.10 Document topic matrix 17
5.11 Text to speech flow diagram 20
5.12 Relevant websites visited by the user 22
5.13 Most relevant website based on selected topics 22

7.1 Output for speech to text 39
7.2 Input for text extraction from image 40
7.3 Output for text extraction from image 40
7.4 Output for autocorrect 41
7.5 Output for summarization 42
7.6 Output for the topic modelling 42
7.7 Output for the most relevant website based on selected topics 43
7.8 Output for relevant websites visited by the user 43

ABBREVIATIONS

BGR	Blue, Red, Green
HSV	Hue, Saturation, Value
openCV	Open Source Computer Vision
ISRI	Information Science Research Institute
API	Application Programming Interface
CMU	Carnegie Mellon University
FLAC	Free Audio Lossless Codec
ASR	Automatic Speech Recognition
STT	Speech To Text
TTS	Text-to-Speech
SAPI	Speech Application Programming Interface
SDK	Software Developer's Kit

CHAPTER 1

INTRODUCTION

With the constant creation of digital and print media, we are made to come across volumes of data every day. This puts over 285 million people who suffer from some kind of visual disabilities and the 10 percent of children affected by reading disabilities at a disadvantage[18] [15]. Our system takes into consideration the problems people with visual impairment and reading disorders have to go through to read and understand textual documents. The system assists these people by reading out any scanned or digital documents which help them in understanding information through auditive assistance. Apart from reading the whole document, the system also allows the user to read out a summarized version of the entire document after the digital text undergoes auto correction to rectify any errors. During the summarization of the digital text, the most important sentences from the text are picked up to build a summary of 5 lines.To determine the topics available in the document, Latent Dirichlet Allocation (LDA) is used. LDA is way of identifying topics automatically by representing a corpus as a mixture of topics that generate words based on the probability distribution assigned to them. This is done when the user interacts with the system by naming the topics they would want to search upon. A web search is automatically done based on the user's choice of one or more topics, and the most relevant web page is opened and read out to the user. Apart from a web search, the user's web browser history is also searched to provide the websites the user has previously visited which are relate to the topics previously selected. The input taken from the user is through speech, so as to make sure that this system can be used conveniently by people who are visually impaired or have reading disabilities.[10][19]

CHAPTER 2

LITERATURE SURVEY

2.1 Overview of Tesseract

Tesseract is an open source Optical Character Recognition (OCR) Engine that was developed by HP. It is considered to be an accurate and free open source OCR engine available and is now maintained by Google. The first step in Tesseract is line finding and word finding. This step also includes baseline fitting and fixed pitch detection and chopping. This is then followed by word recognition where joined characters are chopped off and broken characters are associated together. Following that a static character classification and a linguistic analysis is done.[17] After Tesseract was developed, there were various systems that used it as their OCR engine. One such system is the Tesseract based Optical Character Recognizer (OCR) for Bangla script. Large amount of training data is prepared and a large experiment is performed with their data to seek out the right combination. While testing, Tesseract specification is followed and an intermediate image is made from the segmentation information. An efficient post processor is included to better the performance and reliability of the software. This application is developed for both windows and linux environment.[9]

2.2 Processing

Useful information for automatic explanation, indexing, and structuring of images is contained in textual data present in pictures. Detection, tracking, extraction, and recognition of the text from a given image involves extraction of this data. The problem of automatic text extraction being extremely challenging in the computer vision research area.[13]

The variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background are the reason behind it. Text

Recognition involves a computer system designed to convert images of typewritten text into digital text or to translate images of characters into a standard encoding scheme representing them. The system scans the textual image and then converts it into grayscale. Later, the image is then converted to binary image. Preprocessing such as noise reduction, skew correction etc is done and lastly segmentation is done by spreading lines from textual image. This proposed algorithm is a way of solving the problem to offline character recognition. This system is very efficient to extract all types of bimodal pictures including illumination and blur.[5][3]

Throughout the years, as technology has evolved, there have been multiple advancements to make everyday tasks easier for visually impaired people. Some of them are as given below:

2.3 Wearable Face Recognition System For Visually Impaired People

One of the biggest challenges faced by visually impaired people is the problem of face recognition. This is overcome by using a Microsoft Kinect sensor as a wearable device which does face detection and also generates an audio mapped with the identified person, virtualized at his estimated 3-D location. For, this hardware platform should be portable and cost effective to reach the majority of visually impaired people. The Assistant uses algorithm such as PCA and LDA which are the best ones at providing face images. A Kinect wearable face recognition system uses RBG-D image, as an efficient algorithm to detect faces in all direction. The system can be improved by including the ability to recognize human emotions such as happiness, sadness and other various emotions.[8][11]

2.4 Advanced GPS and GSM Navigation for Blinds

In this system, a GPS-based advanced navigation device is developed to help the blind people with audio assistance in International language to navigate the environment

without asking anyone. It finds out optimal paths to various landmarks near the university area by using heuristic search for the following waypoints. The functioning of this system includes three parts; first the location based speech recording part, second, the navigation of the blind individual by the signal from the GPS receiver and third, sending of the location of the Blind person to his relative's GSM mobile no. The system can be improved by letting the GPS system applicable to indoor surroundings or closed areas.[4]

CHAPTER 3

SYSTEM ANALYSIS

The system is expected to help visually impaired people and those with reading disabilities with everyday tasks such as reading. We make sure that the system operates completely through speech so that the system is capable of reading out not only the documents, but also the summary, topics available in the document for the user. The system makes sure that the user is not expected to read or write anything at any given point. To develop such a system, we use Python 3.6 and the various packages available in Python such as python module win32com is used to make the system talk to the user through speech. OpenCV3, is a real time computer vision based library of programmed functions, is used for preprocessing of the image. We also use Tesseract, a OCR Engine maintained by Google for performing various image processing operations internally before actually extracting the text from the image. Lastly, operating system Windows 10 was used for the development of this system. Hardware components such as 8GB RAM, 256GB Hard Disk and Intel Core i3 Processor at 2.5GHz was used for the development of the system. The system is required to be accurate during extraction of image from text and should be able to compute the precise summary for the text extracted. The topics determined after using topic modeling should be right and actually exist in the text. The system is also required to comprehend the speech of the user and display the accurate output according to the users input. The system should also be extensible in nature and be able to work on multiple platforms so that there is a possibility of future growth. The system should also be well documented and have an user manual so that it can be maintained in the right order even after continuous use.

CHAPTER 4

SYSTEM DESIGN

The user is made to speak to the system by denoting the image it wants the system to extract the text from. This is done through speech for the convenience of the visually impaired and people with reading disabilities. Once the system hears the image to be opened, it searches the file directory for the image. The image is opened to the system and python is used for processing of the image. The text is extracted from the image and this text is then made to go through autocorrection to reduce any kinds of error. Once the text is completely extracted, the text can be read out to the user if he chooses to do so. Otherwise the user can choose if they want to hear the summary of the text or not. Summarization of the extracted text is done and read out to the user. Topic Modeling is then done on the text to extracts the topics that are available in the text. These topics are then read out to the user so that the user can understand what all topics the text is composed off. Once that is done, the user can search more on the topics that are generated. They can do that by reading out the combination of topic(s) they want to search and a Google search for those topics are done. The websites which are relevant to those topics are displayed and the most relevant website is opened and displayed to the user. Moreover, the Firefox browser history is searched and the most relevant searches made are displayed along with the number of times the site has been opened before. This is displayed in ascending order of hits. The system design is represented in Figure 4.1(given below).

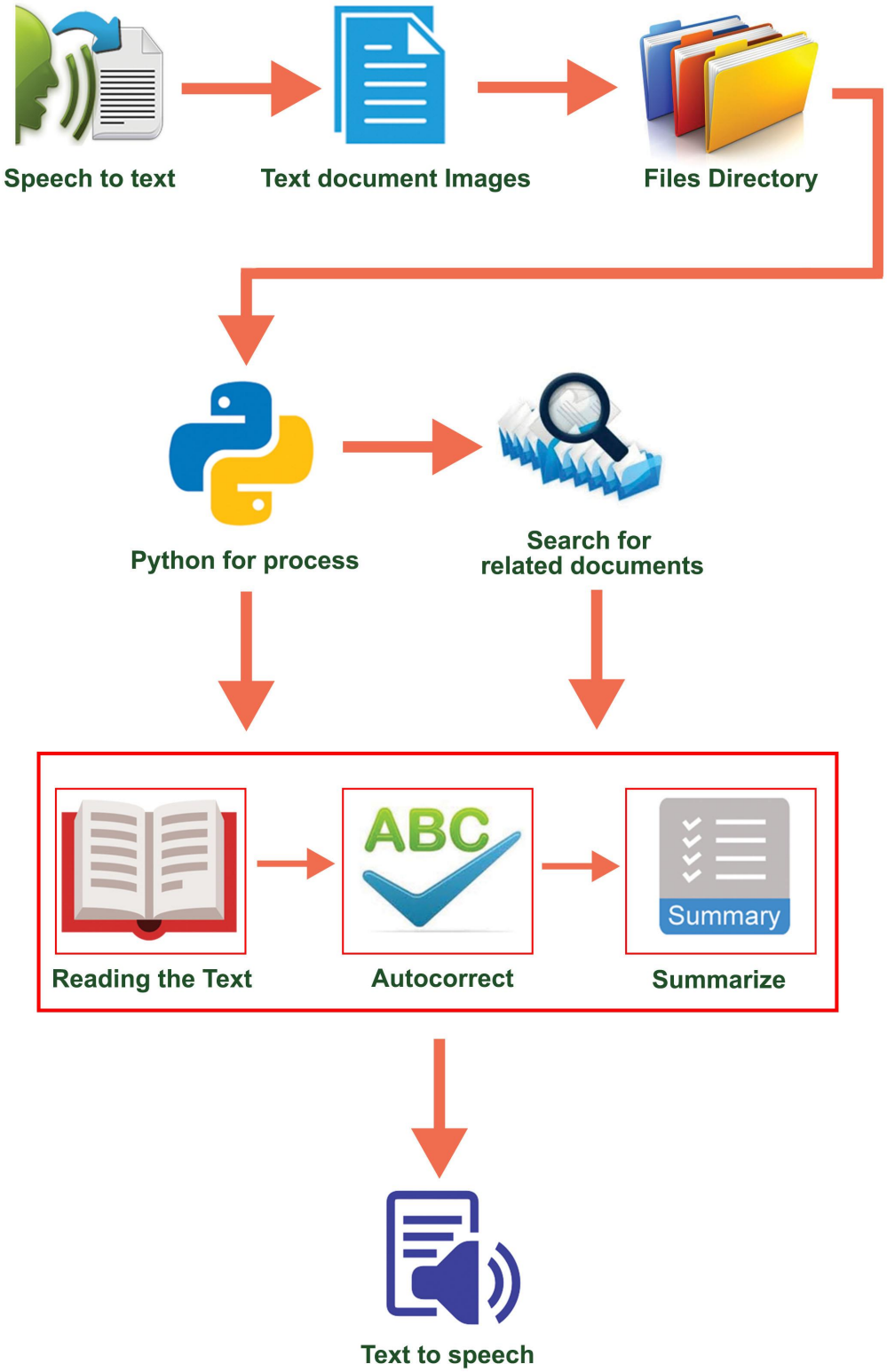


Figure 4.1: Flowchart of system design

CHAPTER 5

METHODOLOGY

5.1 Image Processing

Image processing is the technique applied to an image to clean it before sending them off for further steps. It is done to make the program better understand what it's dealing with as it doesn't look at images we do with our eyes. The program only sees a huge amount of integer values in a 2D array, each denoting a single pixel in the image, so it's very important that it focuses on the necessary ones. Sometimes simple image processing may involve cropping an image other times it may involve performing some logarithmic functions which mathematically enables the program to correctly make estimates such as tumor size. Image processing thus has many methodologies which can be applied with respect to how the image is to be used. These include geometric transformations (scaling, rotation etc), changing colour spaces (Blue, Red, Green (BGR), Hue, Saturation, Value (HSV)), thresholding, smoothing, gradient transformation, edge detections, contour mapping, image segmentation etc.[12]

In our project, we make use of certain methods which help detect the text better in the image and we apply the techniques selectively for this purpose. The python package used for this task is Open Source Computer Vision (openCV) which includes reading and performing said methods on the images. OpenCV, real-time computer vision, is a module of many functions. It was originally developed by Intel, and then later supported by Willow Garage. OpenCV was finally acquired by Intel in 2006. OpenCV is a cross-platform package and made free for use under the open-source BSD license. OpenCV can be used with the deep learning frameworks TensorFlow, Torch/PyTorch etc.

The image processing techniques used are:

5.1.1 Gray Scale Conversion

Normal images are in RBG format, that is each pixel has three sets of values, each value contributing to make it as red, green and blue light and adding together in number of ways to generate a broad range of colors. Grayscale values are in range 0-255 and each value is a mean of the three RGB values for that pixel. Grayscale is preferred because of many reasons such as

Image compression: normal image has 3 values for each pixel but gray scale has only one thus ends up saving space.

Signal to noise: To identify important features in an image, color does not contribute as much in image processing.

Speed: Since the number of values to be dealt with reduces, processing of the image is done faster.

Figure 5.1: RGB to grayscale conversion

5.1.2 Thresholding

It is a process in which if a pixel value is greater than a certain limit, specified by user or statistically generated, it is allocated one value (maybe white), otherwise it is assigned another value (maybe black). This basically helps in binarization of an image which results in way better results as opposed to an image having multiple values. The text in the image becomes easy to detect and read. After converting to grayscale, the values of image will range in 0-255. But after thresholding, they will be either of two values. This helps a lot as the text part of the image will be assigned one value, that is black

and the background the other, white. We apply a specific technique of thresholding that is OTSU's binarization.

OTSU's Binarization: In a bimodal image(bimodal image is an image which has two peaks in its histogram) the threshold value is taken as middle of the two peaks. It automatically calculates a threshold value for a bimodal image.

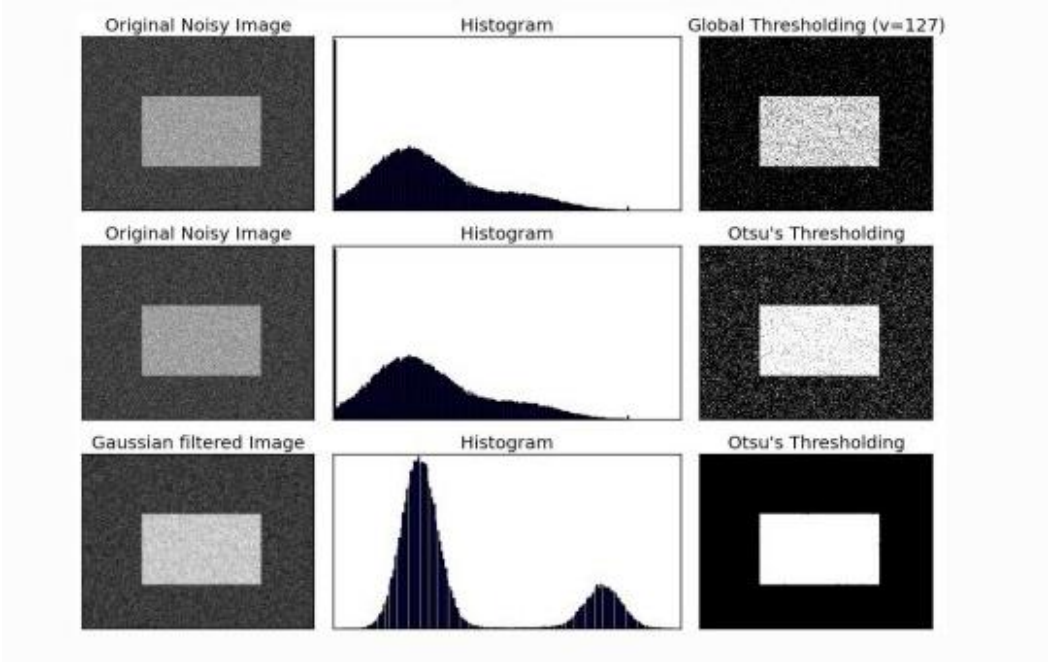


Figure 5.2: OTSU's Binarization

It minimizes theÂăweighted within-class varianceÂăgiven(1.1) by value of which occurs in between two peaks in a way that variances to both classes are least.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{1.1}$$

where

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_1(t) = \sum_{i=t+1}^{I} P(i) \tag{1.2}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_1(t)]^2 \frac{P(i)}{q_2(t)}$$

Figure 5.3: OTSU's binarization equation

Erosion: It is a type of Morphological transformation. These are image shape based operations. It applied to binary images. It needs two things, an image and the other is calledÂăkernelÂăwhich decides the nature of operation. What erosion does is it eats away the boundaries of foreground object thus making its area more. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise, it is eroded (made to zero). All the pixels near boundary will be discarded depending upon the size of the kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. Since our background is white and text (foreground) is black, it makes the white part surrounding the text erode thus increasing the space occupied by the text. It is useful for removing small noises, make the text appear bolder etc.[6]

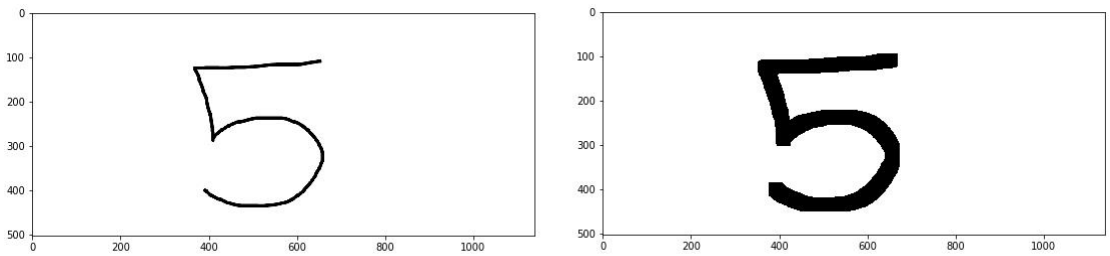


Figure 5.4: Erosion

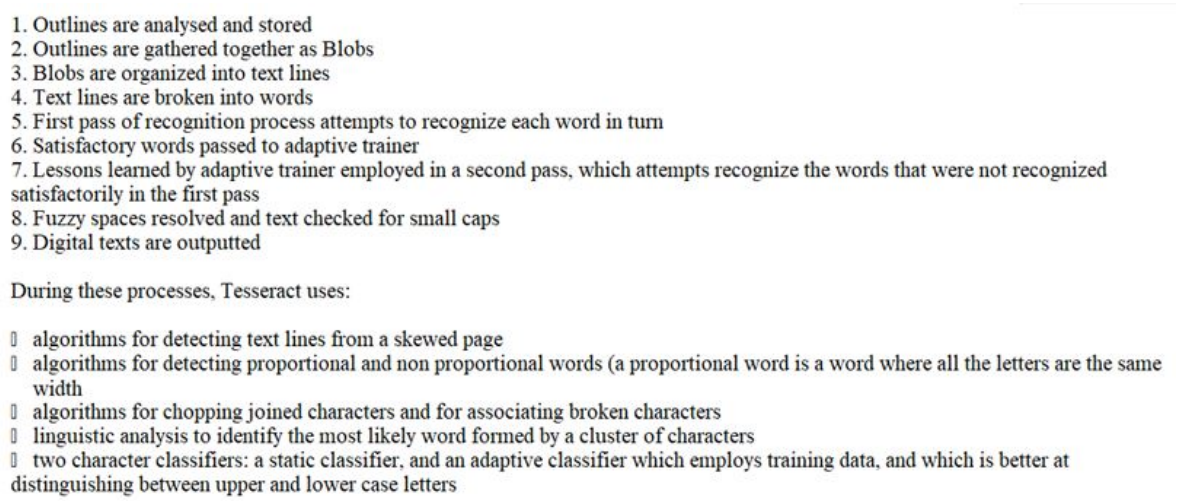
5.2 Image to Text

The image after going through processing is ready to be extracted text from. The processing makes it easier to detect the text. The image is sent to Tesseract using a python module Pytersseract which is a python wrapper.[7]

5.2.1 Tesseract

Tesseract is an OCR engine compatible with various operating systems(Linux, Windows and MAC OS X). The conversion of scanned images containing written text or symbols, for example from a page of a book, into text or data is what OCR technology

does. It is done to understand orÂăaltered byÂăemploying aÂăcomputer program. It was originally built in HP back in 1994. A lot of its code was written in C and then converted to C++. Initially, it was only built for English language detection but now supports many like French, German, Chinese, Tamil, Hindi etc. In 2005, HP released Tesseract's unaltered code to the Information Science Research Institute (ISRI) and it was issued as open source. It is a free software which from 2006 has been developed by Google. It is thought of to be the most precise open source engine. It is usable as a backend and can be integrated into environments like Python. The input to tesseract must be preprocessed for optimal results.[16]


1. Outlines are analysed and stored
2. Outlines are gathered together as Blobs
3. Blobs are organized into text lines
4. Text lines are broken into words
5. First pass of recognition process attempts to recognize each word in turn
6. Satisfactory words passed to adaptive trainer
7. Lessons learned by adaptive trainer employed in a second pass, which attempts recognize the words that were not recognized satisfactorily in the first pass
8. Fuzzy spaces resolved and text checked for small caps
9. Digital texts are outputted

During these processes, Tesseract uses:

- algorithms for detecting text lines from a skewed page
- algorithms for detecting proportional and non proportional words (a proportional word is a word where all the letters are the same width
- algorithms for chopping joined characters and for associating broken characters
- linguistic analysis to identify the most likely word formed by a cluster of characters
- two character classifiers: a static classifier, and an adaptive classifier which employs training data, and which is better at distinguishing between upper and lower case letters

Figure 5.5: Steps of working of tesseract

The above processes ensure that Tesseract is highly precise when recognizing texts from languages which are currently supported. Results from The Fourth Annual Test of OCR Accuracy where, for example, Tesseract demonstrated a Word Accuracy of 97.69% with a sample of English newspapers. Since these tests, the Tesseract development team at Google claim to have improved Tesseract's general results by 7.31% for Tesseract version 2.0.

5.2.2 Pytesseract

Python-tesseract is a python wrapper for Google's open source Tesseract-OCR. Using python as an environment to read the image, it recognizes the text which is embedded in an image. It features as a stand-alone callable script to tesseract. All kinds of im-

ages which are supported by PIL, including png, jpeg, bmp etc can be passed through pytersseract whereas tesseract-OCR only supports tiff and bmp by default. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

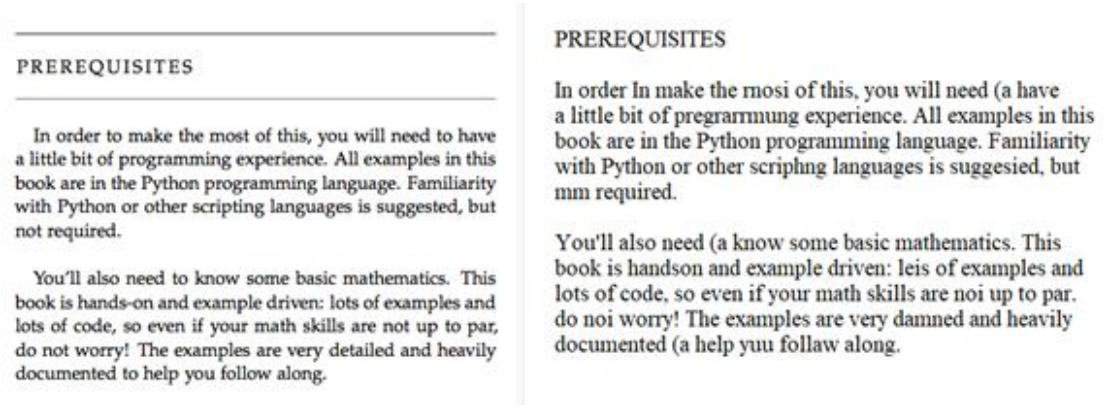


Figure 5.6: The image on the left is converted to text using pytesseract in python

Functions of pytesseract include:

- image_to_string: This function returns the output of a Tesseract OCR run on the image to string.
- image_to_boxes: This function returns result having recognized characters and their respective box boundaries.
- image_to_data: This function returns output having box boundaries, confidences, and other information Tesseract OCR observes.

5.3 Autocorrection

It can said to be a software function that on its own makes or suggests corrections for errors in spelling or grammar which occur in any passage. It is also called as Text replacement or replace-as-you-type which is an information validation function usually seen in word processors and text editing interfaces for smartphones, laptops and tablet computers. Its main purpose is as part of the spell checker to fix common spelling or typing errors allowing user to save time by reducing his load. Another function is to automatically edit text and place special characters.

After extracting text from the image we are left with somewhat a raw output. This has many spelling mistakes which creep into from the conversion of image to text pro-

cess. To further improve upon the accuracy of the text, we apply autocorrect on it. To do this we use autocorrect python package.

5.3.1 Autocorrect

Autocorrecting involves taking each word from the text and performing a check on it. The autocorrect package has a spell function which when given a word checks if it's correctly spelled or not. If it is, the word is returned as such otherwise, it is corrected and then returned. Thus this ensures that all the words in the output are meaningful English words and not junk as sometimes returned by Tesseract.

```
>>> from autocorrect import spell
>>> spell('HTe')
'The'
```

Figure 5.7: Autocorrection

5.4 Text Summarization

To summarize the text extracted from the image first preprocessing is done to make sure that the text has no stop words or punctuations. Then frequency of each word in the preprocessed text is calculated. The frequency of each word is then divided by the maximum value of frequency by any word which has been computed. If the calculated frequency (which will now be in between 0 and 1) falls under a particular range then this word is accepted, else the frequency value of the word is deleted. The calculated frequency or the new frequency of each word in the text acts as the score of the particular word. Lastly, the text is taken and the scores of the words (words which do not fall under the predefined range or are stop words do not have a score assigned to them) is added in every sentence. This helps in computing the ranking of the sentence. The sentences with the highest ranking are considered for the summary of the original text.

5.4.1 Initializing The Range And Stop Words

A range is initialized the and the words that come under stop words, punctuations is stored in a variable. In this case, the minimum range is 0.1 and the maximum range is 0.9. After the new frequency is calculated, the word is given a score only if the new frequency falls under that particular range. Otherwise, the rank of the word is discarded.

5.4.2 Computing Frequency of Each Word

The text is first broken down into individual sentences and each sentence is broken down into individual words. This is done by using the Nltk library available in python. By using a default dictionary, the order of the dictionary is maintained and the frequency of the words are computed and stored along with the word itself. The frequency of each word which is not a stop word is calculated. The highest frequency of any word in the list of words computed and is divided from the actual frequencies of the words calculated before to get the new frequency between 0 and 1. The new frequency or score of each word in the list of words in checked to see if it falls under the predefined range, between 0.1 and 0.9. If the frequency of the word does not fall in this range, then the frequency value of the word is deleted and no rank is given to the word.

5.4.3 Generating The Summary

The number of sentences needed to make the summary, n is provided by the user and is passed on to the function summarize. The code checks is the number of sentences in the text itself is greater that the number of sentences the user wants the summary to be. Only if the lines in the sentence is greater, the system continues to generate the summary. The assert function available in the code checks for the same. The ranking of each sentence is computed by adding the score of each word in the list of words. N of the largest ranked sentences are computed by using the largest function in the heapq library available in python. The ranking of each of the sentences are stored with the sentences in a default dictionary. These sentences are printed to build the summary of the text extracted from the image.

5.4.4 Printing The Summary

The returned summary is then printed according to the number of lines of summary needed by the user. This summary is then read out to the user by the system if the user chooses the option of wanting to listen to the summary.

5.5 Topic Modelling

Topic Modeling is done to generate the topics available in the text extracted from the image. To do so, we use Latent Dirichlet Allocation (LDA). LDA is a way of automatically discovering topics which are available in the list of sentences in the text.It assumes that documents are made from a mix of topics. These topics then generate words based on their probability distribution. LDA later backtracks and finds out the topic which would originally create the document. Any corpus can be represented in a document-term matrix. Figure 5.8 shows the a corpus of N documents such as D1, D2, D3... Dn having the size of the vocabulary of words such as W1, W2, W3... Wn.

	W1	W2	W3	Wn
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
Dn	1	1	3	0

Figure 5.8: Document term matrix

LDA then converts this matrix into two lower dimensional matrix- a document-topic matrix and a topic-term matrix. Let these two matrices be called M1 and M2 matrix. M1 represents a document-topic metrix whereas M2 represents a topic-term matrix. Figure 5.9 represents M1 and Figure 5.10 represents M2. The main aim of LDA is to

	W1	W2	W3	Wm
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K	1	1	0	0

Figure 5.9: Topic term matrix

	K1	K2	K3	K
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
Dn	1	0	1	0

Figure 5.10: Document topic matrix

improve these distributions. It Iterates through each word in each document available in the corpus and tries to replace the current topic by a new topic to a given word with a probability. For each topic, two probabilities are calculated. These probabilities can be called p1 and p2. P1 can be defined as p(topic t / document d). In words, the amount of words in document d that are currently have topic t. P2 can be defined as p(word w / topic t). In words, the amount of topic t assigned to over all documents that come from the word w. The current topic assigned to a word is continuously updated with a new probability which can be denoted by P. P is the product of p1 and p2. After multiple iterations, a steady state is said to have achieved. In this state the distributions-document and topic as well as topic and term are good and does not change. This point is considered as the convergence point of LDA.[2]

5.5.1 Parameters of LDA

There are two parameters of LDA and they are alpha hyper parameter and beta hyper parameter. Alpha hyper parameter represents document topic density. When the value of Alpha is high, the documents tend to be composed of a greater amount of topics.

Other way around, the lesser the alpha value, the documents are composed of lesser topics. Beta hyper parameter represents the topic word density. Higher the parameter of beta, topics tend to be composed of larger number of words. Lower the beta value, they tend to be composed of few amount of words.

5.5.2 Cleaning and Processing

Cleaning is an important task before topic modeling is done and in this step all stop words, punctuations are removed and the text is normalized. This is done using the python library nltk to import stop words and lemmatizer to normalize the word. String library is imported to help in removing the punctuations in the text.

5.5.3 Preparing Document Term Matrix

The corpus is then changed in a document term matrix. This is done by using a python library known as gensim which is used to handle data which is in form of text. It is also efficient and scalable.

5.5.4 Running The LDA Model

After this we need to create an object for LDA model created. This object will be used to train on Document and Term matrix. The training on the matrix requires multiple parameters as input such as the alpha and beta value. The parameters have been explained before. The gensim library is useful for two reasons. Firstly to help in the model estimation of LDA and secondly to assist in topic distribution, even in unseen documents.

5.6 Speech To Text

Speech recognition develops methodologies and technologies that allows the recognition and translation of spoken communication into text by computers, is the inter-

disciplinary sub-field of linguistics. It's conjointly called Automatic Speech Recognition (ASR), computer speech recognition or Speech To Text (STT). It incorporates information and analysis within the linguistics, applied science, electrical engineering fields, and computer science fields. As the project is aimed to help visually impaired people, the best way to make them interact with the system would be using speech. They can issue commands to select, convert and search about an image. They are given audio prompts to speak up. To include this functionality, a python package called SpeechRecognition was used.[10][19]

5.6.1 Speechrecognition

This python tool is used to performs speech detection and recognition with help of several engines and Application Programming Interface (API)'s, both online and offline. This supports Carnegie Mellon University (CMU) sphinx, Google Speech Recognition, Wit.ai and many more.

This works by first enabling the microphone to record audio for a short duration. During this the microphone records the command spoken by the user and the program saves it as a Free Audio Lossless Codec (FLAC) file.

It performs speech recognition on "audio_data" using the Google Speech Recognition API. The Google Speech Recognition API key is specified by "key". If not specified, it uses a generic key that works out of the box.

The recognition language is determined by "language", an RFC5646 language tag like "en-US" (US English) or "fr-FR" (International French), defaulting to US English. The audio data is used to make a key and is then sent to http://www.google.com/speech-api/v2/recognize? for recognition along with the language type. Google returns the result after performing speech recognition. It raises a "speech_recognition.UnknownValueError" exception if the speech is unintelligible or raises a "speech_recognition.RequestError" exception if the speech recognition operation failed, if the key isn't valid, or if there is no internet connection.

5.7 Text To Speech

Artificially produced human speech is known as speech synthesis. A speech computer, also known as a speech synthesizer is a computer system used for the same purpose, which can be applied in software as well as hardware systems. A Text-to-Speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

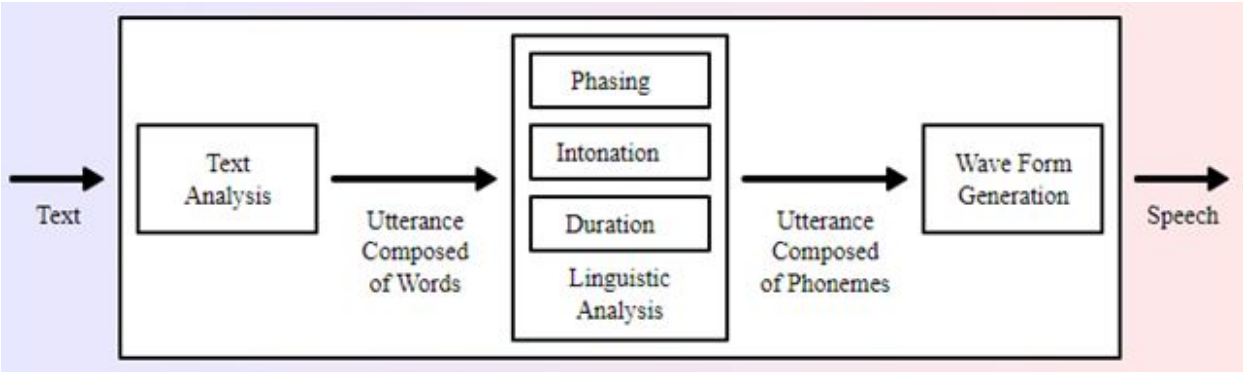


Figure 5.11: Text to speech flow diagram

Since the visually impaired people may have difficulty to read what is written on the image, we first extract its text and then perform autocorrect on it. After these steps the user is asked if he wants to hear out the text, summary or the topics. This way the user can interact with the system even without having to read. Not only this, whatever the user speaks as a command to the system it is read aloud so that there is no confusion and the system is simple and easy to use and understand. It makes the system a hands free model. There are several APIs available to convert text to speech in python. One of such APIs available in the python library commonly known as win32com library. It utilizes Microsoft speech engine.

5.7.1 Windows Search Engine

Speech Application Programming Interface (SAPI), the term for windows speech engine is an API developed and designed by Microsoft. Use of speech recognition and speech synthesis within Windows applications is provided as a feature.

5.7.2 Win32com

To use the windows speech engine, a python package called win32com is used. It has a Dispatch method which when passed with the argument of "SAPI.SpVoice" interacts with the Microsoft Speech Software Developer's Kit (SDK) to speak what is given to it as input. This enables to integrate TTS functionality in a python program.

5.8 Accessing Internet And Web Browsing History

After converting an image to text and providing user its summary and topics, we further provide two more functionalities. They are:

5.8.1 Accessing Internet

After the topics have been made known to the user, he is made to select the topics he wants to gather more information on. Using these topics provided by the user through speech, the program performs a google search. It then makes a collection of top five hits on the search and opens the first URL in the browser. This was the user does not necessarily need to open, type and then search for more information. This enables him to easily access internet. This is done by the webbroswer package of python.

Webbrowser: The module to display web based documents to users, provides a high-level interface and easy to use functions. In usual cases, only calling the open() function from this module will open a tab with the address provided in the function. It supports Firefox, chrome, safari etc.[14]

5.8.2 Searching Web Browser History

After the user provides the topics, the programs opens up the firefox databse in the computer, which stores information like web browser history, bookmarks etc. From this the history is accessed and brought back. This contains the URL which has the topics in it, which means that this site was opened by the user previously and might be

of help currently. The result is cleaned by sorting the URL's by number of times they were opened in a decreasing order that is, most visited sites are shown at the top. This is done using python's sqlite3 package.

sqlite3: SQLite built on C, is a library that does not need a separate server process, provides a disk-based database. Database access can be done using a nonstandard variant of the SQL query language. Applications can also use SQLite for internal data storage. The sqlite3 module was written by Gerhard Haring.[1]

```
which all topics would you like to search?(enter comma separated):
1,2
which all topics would you like to search?
Listening.....Say something!


recognizing........

you said: feeding system

searching....


Following are the most relevant sites you visited based on the topics you selected
system is in site  ('https://en.wikipedia.org/wiki/Operating_system', 4)
operating is in site  ('https://en.wikipedia.org/wiki/Operating_system', 4)
system is in site  ('https://www.webopedia.com/TERM/O/operating_system.html', 1)
operating is in site  ('https://www.webopedia.com/TERM/O/operating_system.html', 1)
system is in site  ('https://www.google.com/url?
sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwi4pJavovvZAhVENI8KHXBGBYQQFghGMAM&url=https%3A%2F
%2Fwww.webopedia.com%2FTERM%2FO%2Foperating_system.html&usg=AOvVaw0YZM4Vpvd4zqFNiE3aq_i', 1)
operating is in site  ('https://www.google.com/url?
sa=t&rct=j&q=&esrc=s&source=web&cd=4&cad=rja&uact=8&ved=0ahUKEwi4pJavovvZAhVENI8KHXBGBYQQFghGMAM&url=https%3A%2F
%2Fwww.webopedia.com%2FTERM%2FO%2Foperating_system.html&usg=AOvVaw0YZM4Vpvd4zqFNiE3aq_i', 1)
system is in site  ('https://www.google.com/search?q=operating+system&ie=utf-8&oe=utf-8', 1)
operating is in site  ('https://www.google.com/search?q=operating+system&ie=utf-8&oe=utf-8', 1)
system is in site  ('https://en.wikipedia.org/wiki/System_software', 1)
system is in site  ('https://www.google.com/search?q=operting+system&ie=utf-8&oe=utf-8', 1)
```

Figure 5.12: Relevant websites visited by the user

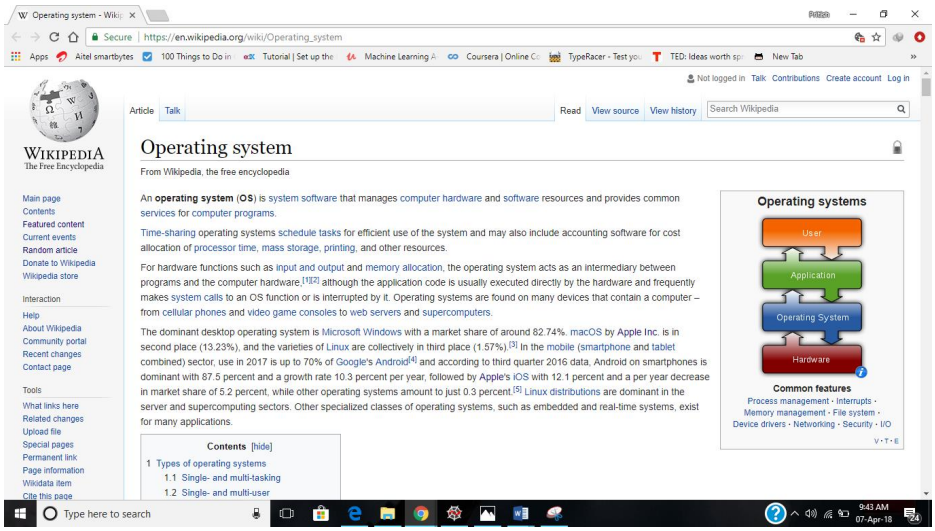


Figure 5.13: Most relevant website based on selected topics

The figure 5.12 shows how the user can provide topics to the program which then opens a website(figure 5.13) which is most relevant to the provided topics and then checks web browser history for sites visited for similar topics and their count

CHAPTER 6

CODING AND TESTING

6.1 Coding

```
# import the necessary packages
# Importing Gensimno

import gensim
from gensim import corpora
import speech_recognition as sr
import webbrowser
import requests

#import os
from nltk.tokenize import sent_tokenize,word_tokenize
from nltk.corpus import stopwords
from collections import defaultdict
from string import punctuation
from heapq import nlargest
from PIL import Image
from googlesearch import search
import pytesseract
import cv2
import os
import numpy as np
from matplotlib import pyplot as plt
import win32com.client as wincl
from autocorrect import spell
```

```
import sqlite3
from nltk.stem.wordnet import WordNetLemmatizer
import string


pytesseract.pytesseract.tesseract_cmd = r'G:\Study\SEM8\
Tesseract-OCR\tesseract.exe'
#pytesseract.pytesseract.tesseract_cmd = r'C:\Users\320005936\AppDat
Lib\site-packages\pytesseract'

# load the example image and convert it to grayscale
image = cv2.imread(r"G:\Study\SEM 8\trial\os1.png")
#image = cv2.imread(r"G:\Study\SEM 8\trial\text4.png")
#image = cv2.imread(r"G:\Study\SEM 8\trial\prisha.png")
#image = cv2.imread(r"G:\Study\SEM 8\trial\os.png")


gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# check to see if we should apply thresholding to
# preprocess the image
ret3,gray = cv2.threshold
(gray,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
#ret,thresh = cv2.threshold(gray,100,255,0)
# make a check to see if median blurring should be done to
# done to remove noise
#gray = cv2.medianBlur(gray, 3)
#kernel = np.ones((1,1),np.uint8)
#gray = cv2.dilate(gray,kernel,iterations = 1)
kernel = np.ones((2,2),np.uint8)
gray = cv2.erode(gray,kernel,iterations = 1)
```

```
fig=plt.figure()
fig.set_size_inches(9,9)
plt.imshow(gray,cmap="gray")
plt.show()


filename = "read_me.png"
cv2.imwrite(filename, gray)


# load the image as a PIL/Pillow image, apply OCR, and
# then delete the temporary file
text_read=""
text = pytesseract.image_to_string(cv2.imread("read_me.png"))
os.remove(filename)

#performing spell check of passage word by word
sents = sent_tokenize(text)
word_sent = [word_tokenize(s.lower()) for s in sents]
for num,a in enumerate(word_sent):
for b in range(len(a)):
if(len(word_sent)>=3):
word_sent[num][b]=spell(word_sent[num][b])
else:
word_sent[num][b]=word_sent[num][b]
#text_read=text_read+" "+word_sent[num][b]
#text=spell(text)

#printing the passage after performing spell check
for sentence in sents:
sentence=sentence.replace("\n"," ")
for word in sentence.split(" "):
```

```
if(len(word)>=3):
print(spell(word),end=" ")
else:
print(word,end=" ")
print(" ")



# show the output images
#cv2.imshow("Image", image)
#cv2.imshow("Output", gray)
#cv2.waitKey(0)



class FrequencySummarizer:
def __init__(self, min_cut=0.1, max_cut=0.9):
self._min_cut = min_cut
self._max_cut = max_cut
self._stopwords = set(stopwords.words('english') +
list(punctuation))

def _compute_frequencies(self, word_sent):
freq = defaultdict(int)
for s in word_sent:
for word in s:
if word not in self._stopwords:
freq[word] += 1
m = float(max(freq.values()))
if freq[word]==m:
print(word)
for w in list(freq.keys()):
```

```
freq[w] = freq[w]/m
if freq[w] >= self._max_cut or freq[w] <= self._min_cut:
del freq[w]
return freq    # return the frequency list

def summarize(self, text, n,sents):

# split the text into sentences
assert n <= len(sents)
word_sent = text
self._freq = self._compute_frequencies(word_sent)
ranking = defaultdict(int)
for i,sent in enumerate(word_sent):
for w in sent:
# for each word in this sentence
if w in self._freq:
# if this is not a stopword (common word), add the frequency
# of that word to the weightage assigned to that sentence
ranking[i] += self._freq[w]
# OK – we are outside the for loop and now have rankings for
# all the sentences
sents_idx = nlargest(n, ranking, key=ranking.get)
# we want to return the first n sentences with highest
# ranking, use the nlargest function to do so
# this function needs to know how to get the list of values
# to rank, so give it a function – simply the
# get method of the dictionary
return [sents[j] for j in sents_idx]

#textOfUrl = text
fs = FrequencySummarizer()
#summary = fs.summarize(textOfUrl, 5,sents)
```

```
summary = fs.summarize(word_sent, 5,sents)

print("\n\n***************** SUMMARY *****************")
count=1
for a in summary:
print(count,". ",end=" ")
count+=1
a=a.replace("\n"," ")
for b in a.split(" "):
if(len(b)>=3):
#print(b)
print(spell(b),end=' ')

else:
print(b,end=" ")
print("")



stop = set(stopwords.words('english'))
exclude = set(string.punctuation)
lemma = WordNetLemmatizer()

#removing stop words, punctuations
def clean(doc):
stop_free = " ".join([i for i in doc.lower().split()
if i not in stop])
punc_free = ''.join(ch for ch in stop_free
if ch not in exclude)
normalized = " ".join(lemma.lemmatize(word)
for word in punc_free.split())
return normalized
```

```
final=[]

for a in word_sent:
temp=""
for b in a:
temp=temp+" "+b
final.append(temp[:-2])

doc_clean = [clean(doc).split() for doc in final]

# Creating the term dictionary of our courpus, where
# every unique term is assigned an index.
 dictionary = corpora.Dictionary(doc_clean)

# Converting list of documents (corpus) into
# Document Term Matrix using dictionary prepared above.
doc_term_matrix = [dictionary.doc2bow(doc) for doc in
doc_clean]

# Creating the object for LDA model using gensim library
Lda = gensim.models.ldamodel.LdaModel

# Running and Trainign LDA model on the document term matrix.
ldamodel = Lda(doc_term_matrix, num_topics=3, id2word = dictionary,

lda=ldamodel.print_topics(num_topics=3, num_words=3)



#tts = gTTS(text=summary[0], lang='en')
#tts.save("good.mp3")
#os.system("mpg321 good.mp3")
```

```
speak = wincl.Dispatch("SAPI.SpVoice")

speak.Speak("do you want to hear summary of the passage?
Yes or No ")
speak.Speak("Enter your choice by typing or speak when
 asked")

choice=input("do you want to hear summary of the passage?
\n1. Yes\n2. No\n\n")
r = sr.Recognizer()
with sr.Microphone() as source:
speak.Speak("Listening.....Say something!")
print("Listening.....Say something!\n")
r.energy_threshold += 280
audio = r.listen(source)

# Speech recognition using Google Speech Recognition
try:
# for testing purposes, we're just using the default API
# key to use another API key, use `r.recognize_google(audio,
#key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
# instead of `r.recognize_google(audio)`
print("recognizing.....\n")
sum_text=r.recognize_google(audio)
except sr.UnknownValueError:
print("Google Speech Recognition could not understand audio\n")
except sr.RequestError as e:
print("Could not request results from Google Speech Recognition
service; {0}".format(e))

print("you said:",sum_text)
```

```
speak.Speak("you said ")
speak.Speak(sum_text)
if(choice=="1" or choice=="Yes" or choice=="yes" or sum_text=="yes"
or sum_text=="Yes"):
for a in summary:

speak.Speak(a)

print("\n\n****************** TOPICS ******************")
lda_d=[]
print("the topics of passage are:\n\n")
speak.Speak("\n\nthe topics of passage are:")
for a in lda:

temp=a[1]
te=temp.split("+")
for a in te:
tem=(a.split("*")[1])
tem=tem.split("\"")[1]
if(tem not in lda_d and len(tem)>1):
lda_d.append(tem)
count=1
for a in lda_d:
print(count,". ",a,end=" ")
count=count+1
print("")
speak.Speak(a)


speak.Speak("would you like more information on these topics?
Yes or  No")
speak.Speak("Enter your choice by typing or speak when asked")
```

```
choice=input("would you like more information on these topics?
\n1. Yes \n2. No\n\n\n")

with sr.Microphone() as source:
speak.Speak("Listening.....Say something!")
print("Listening.....Say something!\n\n")
r.energy_threshold += 280
audio = r.listen(source)

# Speech recognition using Google Speech Recognition
try:
# for testing purposes, we're just using the default API key
# to use another API key, use `r.recognize_google(audio,
#key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
# instead of `r.recognize_google(audio)`
print("recognizing........\n")
topic_choice_text=r.recognize_google(audio)
except sr.UnknownValueError:
print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
print("Could not request results from Google Speech
Recognition service; {0}".format(e))

print("you said:",topic_choice_text)
speak.Speak("you said ")
speak.Speak(topic_choice_text)

choice="1"
if(choice=="1" or choice=="Yes" or choice=="yes" or
topic_choice_text=="yes" or topic_choice_text=="Yes"):
```

```
speak.Speak("which all topics would you like to search?
(enter comma separated)")
ch_par=input("which all topics would you like to search?
(enter comma separated):\n")

#voice input for searching
speak.Speak("which all topics would you like to search?")
print("which all topics would you like to search?")
with sr.Microphone() as source:
speak.Speak("Listening.....Say something!")
print("Listening.....Say something!\n\n")
r.energy_threshold += 280
audio = r.listen(source)

# Speech recognition using Google Speech Recognition
try:
# for testing purposes, we're just using the default API key
# to use another API key, use `r.recognize_google(audio,
# key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
# instead of `r.recognize_google(audio)`
print("recognizing........\n")
topics_text=r.recognize_google(audio)
except sr.UnknownValueError:
print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
print("Could not request results from Google Speech
Recognition service; {0}".format(e))

print("you said:",topics_text)
speak.Speak("you said ")
speak.Speak(topics_text)
```

```
print()
print("searching....\n\n")
new_search=""
for a in ch_par.split(","):
new_search=new_search+" "+lda_d[int(a)-1]
result=[]
#add try catch block here
for j in search(new_search, tld="co.in", num=5, stop=1, pause=2):
result.append(j)

response = requests.get(result[0])
#if(response.status_code==400):
url = result[0]
# Open URL in a new tab, if a browser window is already open.
#webbrowser.open_new_tab(url)

# Open URL in new window, raising the window if possible.
webbrowser.open_new(url)



#searching by voice
new_search_speech=topics_text



result=[]
#add try catch block here
for j in search(new_search_speech, tld="co.in", num=5, stop=1,
pause=2):
result.append(j)

response = requests.get(result[0])
```

```
#if(response.status_code==400):
url = result[0]
# Open URL in a new tab, if a browser window is already open.
#webbrowser.open_new_tab(url)

# Open URL in new window, raising the window if possible.
webbrowser.open_new(url)

#opening firefox database
data_path = r"C:\Users\pritishrawal\AppData\Roaming\Mozilla\
Firefox\Profiles\kdo4gmo4.default"
files = os.listdir(data_path)
history_db = os.path.join(data_path, 'places.sqlite')

c = sqlite3.connect(history_db)
cursor = c.cursor()

select_statement = "select moz_places.url,
moz_places.visit_count from moz_places;"
cursor.execute(select_statement)
#fetching database
results = cursor.fetchall()

#sorting the tuple according to hits
def last(n):
return n[-1]

def sort(tuples):
return sorted(tuples, key=last)

new_res = sort(results)
new_res.reverse()
```

```
#display most relevant browsing history sorted in decreasing
#order of hits
speak.Speak("Following are the most relevant sites you
visited based on the topics you selected")
print("Following are the most relevant sites you visited
based on the topics you selected")
for sites in new_res:
for topic in lda_d:
if(topic in sites[0].lower()):
print(topic, "is in site ",sites)
```

6.2 Testing

Test case ID	Test Case Name	Test Case description	Test Steps			Test Status (P/F)
			Steps	Expected Result	Actual Result	
1	Image to Text	To extract the text from the image	Preprocess-ing image	Image is binarised and noise is removed	Image is preprocess-ed properly	P
			Conversion	The text in image should be accurately read	The text is extracted from image	P

Table 6.1: Table for Test Cases (continued on the next page)

Test case ID	Test Case Name	Test Case description	Test Steps			Test Status (P/F)
			Steps	Expected Result	Actual Result	
2	Auto-correction	To perform spell check on each word extracted from image	To spell check every word	Every word should be correctly spelled	Words don't have spelling mistake	P
3	Summariz-ation	To select the most important sentences which may summarise the text	Find most used words in the document	A dictionary of words is created along with their frequencies	Word list is created after removing all stop words	P
			Using most used words, select most relevant sentence	Score individual sentences based on words used in it	Most relevant sentences are selected	P
4	LDA	To select the words which represent the text	Create document topic and topic word matrix	Formation of topic matrix	Matrix are formed	P
			Calculate probability of each topic and update current word topic with new probability	To assign probability of each topic	Probabili-ties are generated	P
			Iterate until steady state is achieved	Repeat process until steady state achieved and topics selected	Topics are selected and displayed	P

Table 6.2: Table for Test Cases (continued on the next page)

Test case ID	Test Case Name	Test Case description	Test Steps			Test Status (P/F)
			Steps	Expected Result	Actual Result	
5	Speech to Text	To accept and convert the users audio input into text	Take command from user and convert it to text from speech	Users audio input is correctly translated to text	The speech is converted to text	P
6	Text to Speech	To read out instructions and interact with user using speech	Reading the text given	To speak out the content extracted from text and other assistive elements	The text is converted to speech	P
7	Webrowser Search	To perform an internet search for the topics chosen by the user	Accept topics from user	User is able to provide the topics through speech	The topics are correctly accepted and understood	P
			Search the topics and open relevent website	Combine the topics and use internet to search for websites	Correct website is opened and displayed	P
8	Database Extraction	To refer and bring out previous search made by user on same topics	Open webrowser database and extract information	Connect to web browser database and read history to select certain sites relevant to topics given by user	Correctly displays websites previously opened by user	P

Table 6.3: Table for Test Cases

CHAPTER 7

RESULT

7.1 Speech To Text

Whenever the system reads out a question for the user, the user can reply by speech to answer the question. For example, when the system asks out loud if the user wants to listen to the summary, the user can reply with a yes or a no. This answer is heard and recognized by the system and actions are taken depending on the answer given by the user.

Figure 7.1 shows the output on the screen after the user says yes to a question asked by the system.

Figure 7.1: Output for speech to text

7.2 Extracting Text From Image

The user can convert an image such as in Figure 7.2 to text.

Once the text in the image is converted to digital text, the output would look similar to Figure 7.3.

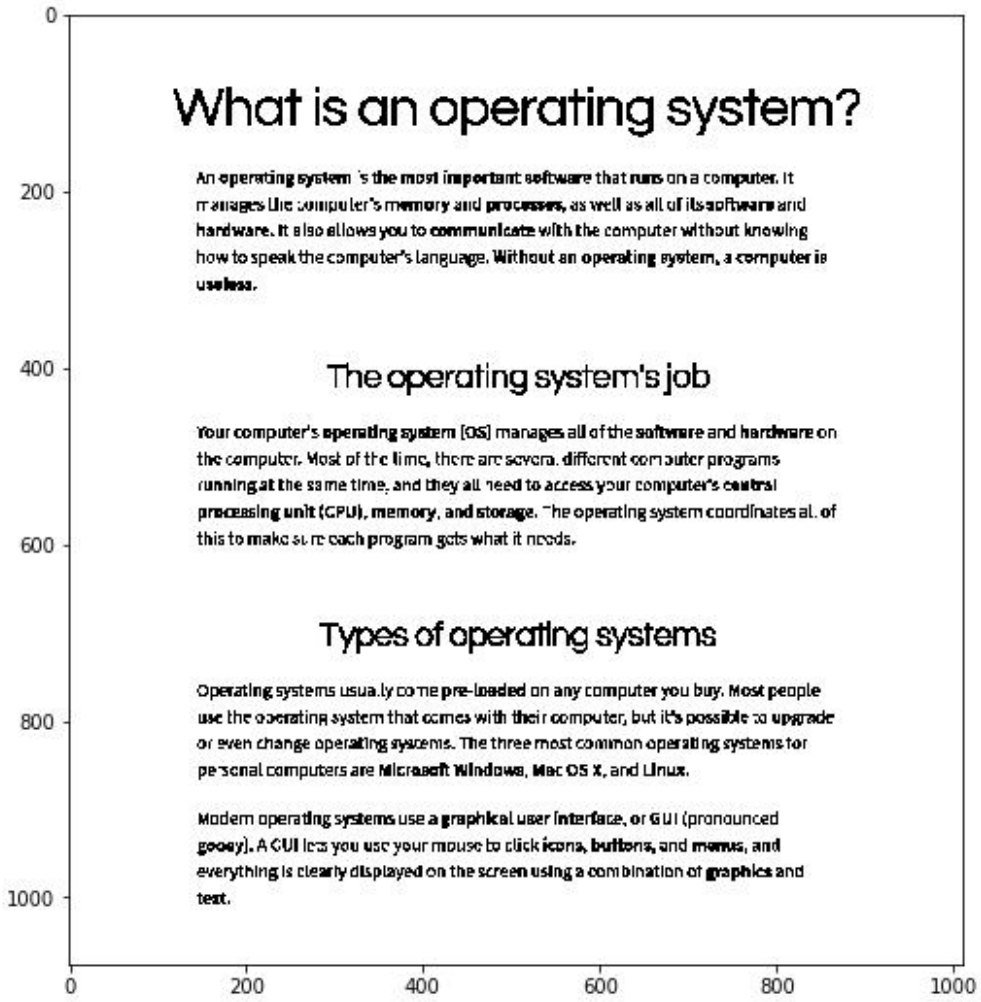


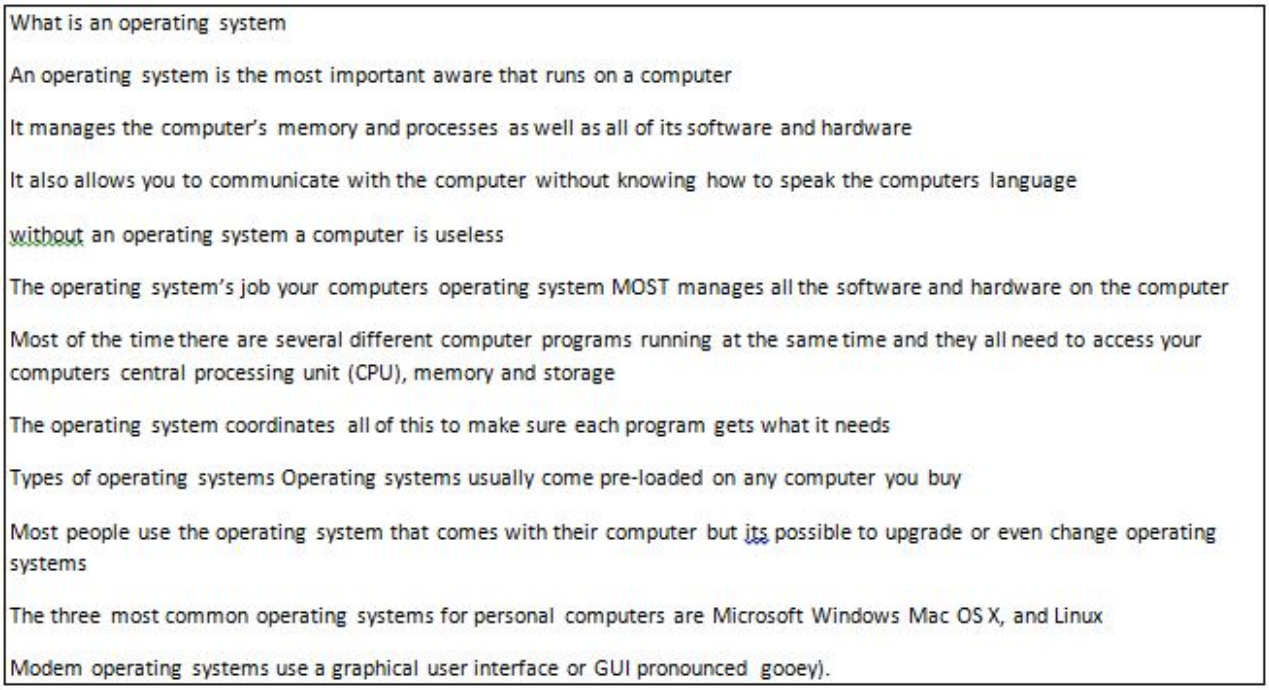
Figure 7.2: Input for text extraction from image

```
Out[16]: "What is an operating system?\n\nAn operating system is the most important mfiware that runs on a computer.
It\nmanages the computer's memory and processes, as well as all of its software and\nhardware. It also allows you to
communicate with the computer without knowing\nhow to speak the computer's language. without an operating system, a
computer is\nuseless.\n\nThe operating system'sjob\n\nVour computer's operating system (OS) manages all ofthe software
and hardware on\nthe computer. Most of the time, there are several different computer programs\nrunning at the same time,
and they all need to access your computer's central\nprocessing unit (CPU), memory, and storage. The operating system
coordinates all of\nthis to make sure each program gets what it needs.\n\nTypes of operating systems\n\nOperating systems
usually come pre-lnaded on any computer you buy. Most people\nuse the operating system that comes with their computer,
but it's possible to upgrade\nor even change operating systems. The three most common operating systems for\npersonal
computers are Microsolt Windows, Mac OS X, and Linux.\n\nModem operating systems use a graphical user interface, or GUI
(pronounced\ngooey)."
```

Figure 7.3: Output for text extraction from image

7.3 Autocorrect

When we extract text from an image, there is a possibility that there are errors during extraction. This errors in text can be minimized by using autocorrection.These errors in text can be minimized by using autocorrection which spell checks individual words before sending it for summarization or topic modeling Figure 7.4 shows the extracted text after undergoing autocorrection.


What is an operating system

An operating system is the most important aware that runs on a computer

It manages the computer's memory and processes as well as all of its software and hardware

It also allows you to communicate with the computer without knowing how to speak the computers language

without an operating system a computer is useless

The operating system's job your computers operating system MOST manages all the software and hardware on the computer

Most of the time there are several different computer programs running at the same time and they all need to access your computers central processing unit (CPU), memory and storage

The operating system coordinates all of this to make sure each program gets what it needs

Types of operating systems Operating systems usually come pre-loaded on any computer you buy

Most people use the operating system that comes with their computer but its possible to upgrade or even change operating systems

The three most common operating systems for personal computers are Microsoft Windows Mac OS X, and Linux

Modem operating systems use a graphical user interface or GUI pronounced gooey).

Figure 7.4: Output for autocorrect

7.4 Summarization

Once the extracted text is autocorrected, the user can choose to summarize the text by communicating with the speech.

Figure 7.5 indicates the output of the extracted text after summarization.

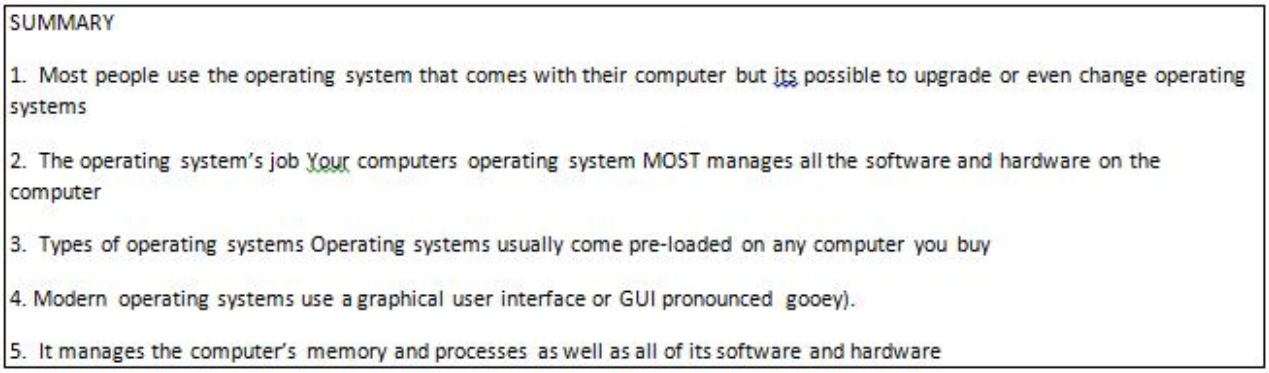


Figure 7.5: Output for summarization

7.5 Topic Modelling

Once the extracted text is autocorrected, the user can choose to summarize the text by communicating with the speech.

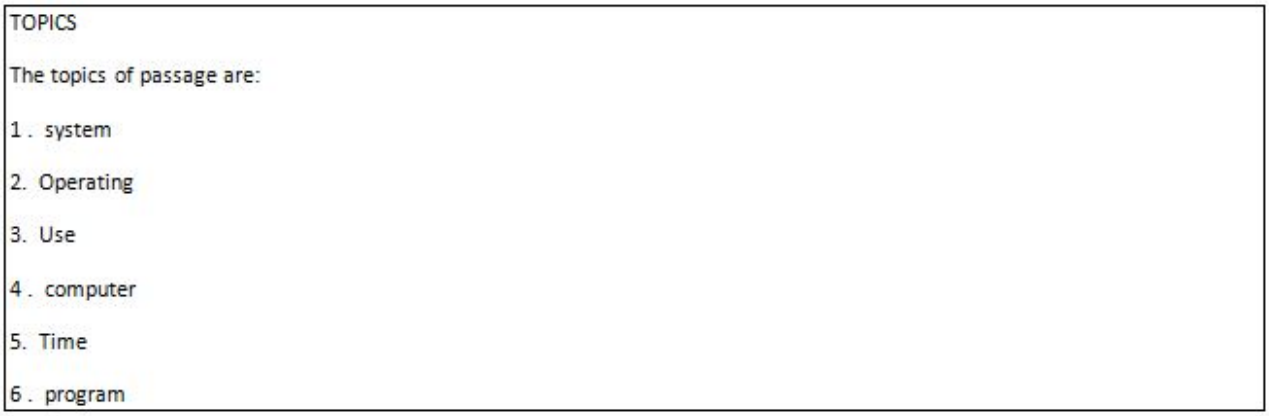


Figure 7.6: Output for the topic modelling

7.6 Text to Speech

After extraction of text from the image or after computing the summary of the text, the output obtained is provided to the user in form of speech. This way a visually impaired person or a person with reading disabilities would not be required to read or see the output obtained.

7.7 Searching Related Document

After the important topics are extracted from the text, the user can list the topics they would want to search upon through speech. By doing so, a google search on the topic(s) is done and the most relevant webpage is opened as shown in figure 7.7. Figure 7.8 indicates the searches relevant to the topic, made by the user previously, extracted by searching the web browser history of the user.

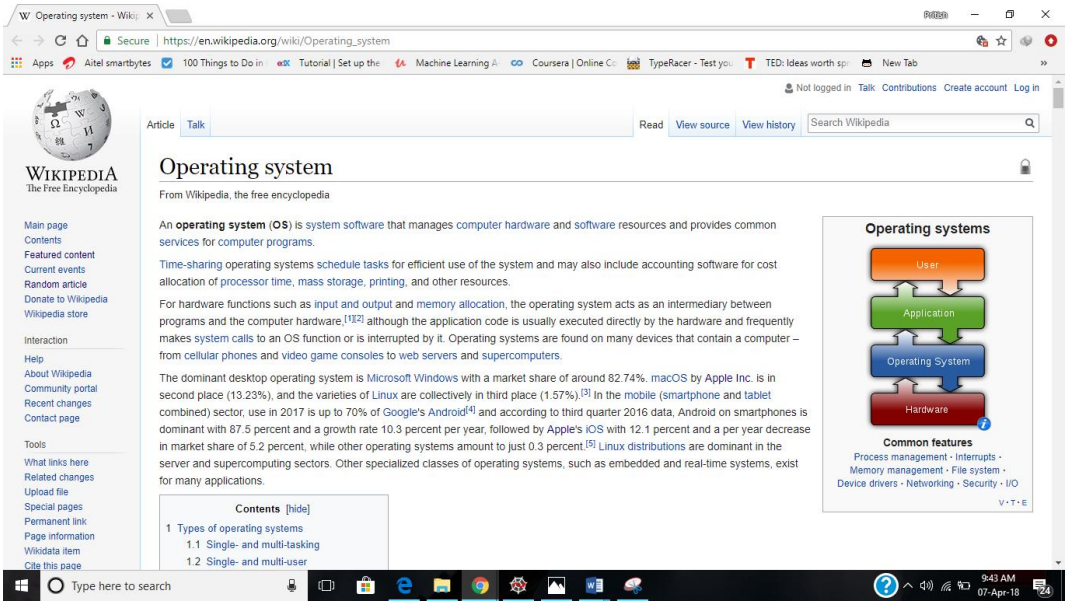


Figure 7.7: Output for the most relevant website based on selected topics

```
Which all topics would you like to search?

Listening.....

recognizing.....

you said: operating system

searching....

Following are the most relevant sites you visited based on the topics you selected

System is in site ('https://en.wikipedia.org/wiki/Operating_system', 4)

operating is in site ('https://en.wikipedia.org/wiki/Operating_system', 4)
```

Figure 7.8: Output for relevant websites visited by the user

CHAPTER 8

FUTURE ENHANCEMENT

There are multiple improvements which can be made to the system which can help make it more efficient, reliable and extensible. One of such future enhancement would be to make sure that the system is platform independent to improve the outreach and mobility of the system. A easy to use User Interface (UI) can also be helpful for the user to communicate with the system. Pre processing technique can be made more accurate according to the image text needs to be extracted from so that it does not have generic rules of preprocessing. As the system communicates with the user through speech, it is important the offline speech is also available so the system can help the user read documents without internet connection. Moreover, the system can be open to comprehending and replying in more languages than English so that the usability of the system increases. Customization of voice can also be allowed depending on the accent and the gender of the voice. Lastly, accuracy could probably be improved significantly with the judicious addition of a Hidden-Markov-Model-based character n-gram model.[17]

CHAPTER 9

CONCLUSION

Advancements in technology should be holistic and cater all kinds of people to provide a uniform base for information exchange. The main aim of the system is to make sure that visually impaired people and those with reading disabilities do not have a difficulty in doing everyday tasks such as reading documents. By using speech as form of a medium for giving information to the user and accepting input from the user, we make sure that people which such disabilities can also use the system at ease and are not left at a disadvantage. The added functionalities of the system helps the user understand and comprehend the document better.

REFERENCES

1. "web.stanford.edu.
2. "www.analyticsvidhya.com.
3. Chowdhury Md Mizan, Tridib Chakraborty, S. K. "Text recognition using image processing." 8.
4. Datta, A. "Advanced gps gsm based navigation system for blinds." 1.
5. Giri, P. S. "Text information extraction and analysis from images using digital image processing techniques.
6. Kaensar, C. *A Comparative Study on Handwriting Digit Recognition Classifier Using Neural Network, Support Vector Machine and K-Nearest Neighbor*.
7. Lili Pan, M. X. "Research on iris image preprocessing algorithm.
8. M. Karthikeyan, Joseph Henry, M. L. M. M. A. V. K. "Wearable chest-mounted navigation system for visually impaired using digital image processing.
9. Md. Abul Hasnat, Muttakinur Rahman Chowdhury, M. K. "An open source tesseract based optical character recognizer for bangla script.
10. Miss.Prachi Khilari, P. B. V. P. "A review on speech to text conversion methods." 4.
11. Poonam Liladhar Agrawal, P. D. D. P. "Wearable face recognition system to aid visually impaired people." 2.
12. Prachi Joshi, Aayush Agarwal, A. D. R. S. S. K. "Handwriting analysis for detection of personality traits using machine learning approach." 130.
13. Pratik Madhukar Manwatkar, S. H. Y. "Text recognition from images.
14. Pratiksha Ashiwal, S.R.Tandan, P. T. R. M. "Web information retrieval using python and beautifulsoup." 4.
15. Rutuja R. Dengale, Bhagyashri S. Deshmukh, A. R. M. S. V. U. "A face recognition and spoofing detection adapted to visuallyimpaired people." 7.
16. S. Anitha, D. "Comparison of image preprocessing techniques for textile texture images." 2.
17. Smith, R. "An overview of the tesseract ocr engine.
18. Smitha.M, T. Ayesha Rumana, S. "Hand gesture based home automation for visually challenged." 2.
19. Su Myat Mon, Hla Myo Tun, A. R. M. S. V. U. "Speech-to-text conversion (stt) system using hidden markov model (hmm)." 4.