IMPLEMENTATION OF AN OPTICAL CHARACTER RECOGNIZER (OCR) FOR BENGALI LANGUAGE

THESIS REPORT

SUPERVISOR**:** DR. MD. KHALILUR RHAMAN

CONDUCTED BY:

MUHAMMED TAWFIQ CHOWDHURY (ID-11101009)

MD.SAIFUL ISLAM (ID-11101061)

BAIJED HOSSAIN BIPUL (ID-11101047)

SCHOOL OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science and Engineering

BRAC UNIVERSITY

Submitted on: 24.08.2015

DECLARATION

This is to certify that this thesis report is submitted by Muhammed Tawfiq Chowdhury (ID-11101009), Md. Saiful Islam (ID-11101061), and Baijed Hossain Bipul (ID-11101047) for the degree of Bachelor of Science in Computer Science and Engineering to the Department of Computer Science and Engineering, School of Engineering and Computer Science, BRAC University. The contents of this thesis have not been submitted elsewhere for the award of any degree or diploma. We hereby declare that this thesis is based on the results found by ourselves and the materials of work found by other researchers are mentioned by reference. We carried out our work by the supervision of Dr. Md. Khalilur Rhaman.

Signature of Supervisor	Signatures of Authors
---------------------------------	---
Dr. Md.Khalilur Rhaman	Muhammed Tawfiq Chowdhury
	--
	Md. Saiful Islam
	--
	Baijed Hossain Bipul

ACKNOWLEDGEMENT

We would like to express the deepest appreciation to our thesis supervisor Dr. Md. Khalilur Rhaman for his guidance, understanding and most importantly his cordial behavior. He continuously guided us instead of his busy schedule. It is an honor for us to say thanks to him, in the light of the fact that we are equipped for building up our understanding in regards to the proposal subject and complete our work which wasn't possible without the assistance of his direction, consolation and consistent backing.

We would also like to thank Muhammad Abdur Rahman Adnan, lecturer BRAC University to help us to solve the GUI related problems.

We additionally need to express gratitude toward S. Mahbub Uz-zaman Ananda, alumni of BRAC University who manages his valuable time his time to answer our issue related email with his recommendation.

We would also like to thank Dr. Mizanur Rahman who demonstrated his own work to us which is related to our thesis and helped us to get an idea.

We would like to thank our team members also to keep a very healthy working environment which actually needed to complete our task. This becomes possible only because of their positive mind and hard work.

We like to thank our parents, for their faith in us and allowing us to be as ambitious as we wanted. It was under their watchful eye that we gained so much drive and an ability to tackle challenges head on.

Last but not the least, in particular we are truly all that much thankful to the Almighty Allah who helps us in every progressions of our proposition work.

ABSTRACT

Optical Character Recognition (OCR) is the process of extracting text from an image. The main purpose of an OCR is to make editable documents from existing paper documents or image files. A number of algorithms are required to develop an OCR. Noise removal, skew identification and correction, segmentation, etc are the different steps of developing an OCR. OCR primarily works in two phases; they are character and word detection. In case of more sophisticated approach, an OCR also works on sentence detection to preserve documents' structures. In this paper, we would discuss the process of developing an OCR for Bengali language. Lots of efforts have been put on developing an OCR for Bengali. Though some OCRs have been developed, none of them is completely error free. For our thesis, we trained Tesseract OCR Engine to develop an OCR for Bengali language. Tesseract is currently the most accurate OCR engine. This engine was developed at HP labs and currently owned by Google. We used a number of software to prepare our training files. Our OCR's library contains 18110 characters and 2617 words. We used 'Solaimanlipi' font in our project. We used 200 input files to test the accuracy of our OCR. We are using the latest 3.03 version of Tesseract for Windows Operating System. For clean image files, the accuracy of our software was as high as 97.56%. It is important to mention that we measured accuracy as the percentage of correct characters and words.

Key Words:
Optical Character Recognition (OCR), Bengali Language, Tesseract, JTessboxEditor, Netbeans IDE

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER 1
INTRODUCTION

1.1 About OCR

In our day to day life, we often need to reprint text with modification. However, in many cases, the printable document of the text does not remain available for editing. For example, if a newspaper article was published 10 years ago, it is quite possible that the text is not available in an editable document such as a word or text file. So, the only choice remains is to type the entire text which is a very exhaustive process if the text is large. The solution of this problem is optical character recognition. It is a process which takes images as inputs and generates the texts contained in the input. So, a user can take an image of the text that he or she wants to print, feed the image into OCR and then the OCR will generate an editable text file for the user which is amendable. This file can be used to print or publish the required text. The software that performs the process is called Optical Character Reader or OCR.

1.2 Motivation

Bengali is one of the most spoken languages of the world. With about 250 million native and about 300 million total speakers worldwide, it is the seventh most spoken language in the world by total number of native speakers and the eleventh most spoken language by total number of speakers. The importance of this language to the countries of South Asia can be noted by the fact that the National Anthem of Bangladesh, National Anthem of India, National Anthem of Sri Lanka and the national song of India were all first composed in the Bengali language. Bengali is written in Sanskrit script. It is very resourceful. However, there have not been so many works on language processing for Bengali as have been for some other languages such as English and Spanish. There are many noteworthy writings that were accomplished in Bengali in the last century. As technology back then was not advanced, these writings were not saved in a digital form. So, it is not possible for a publisher to print them again without typing the entire writings. In this case, an OCR can be a great help for the publishers. An OCR for Bengali would also enable users to make editable files from images that have been generated for distinct purposes. So, we wanted to research on implementation of an OCR for Bengali.

1.3 OCR for Bengali

For Bengali language, there has not been so much work done although in recent time, some projects have been implemented. However, none of them are fully accurate. There have been detached works with no integration. It is also not easy to find much information about developing an OCR for Bengali. Different projects have been done in different methods. Some developers used their own algorithms to develop OCR while some others used existing OCR engines to make OCR. It is not quite easy to develop an OCR for Indic languages like Bengali because of complexity. Bengali, for example, has diverse types of characters and they total to a very huge number. The inter resemblance among the characters makes it even tougher to maintain the accuracy as the OCR may misjudge one character for another. The total number of characters also makes the execution time longer as the scanning process of OCR goes through a very large data set. We preferred to work on an OCR engine for our thesis project. This engine called 'Tesseract' is well tested and it is the most accurate open-sourced OCR engine available. Though there are some limitations, we trained the engine for Bengali for a very intricate and large character set and the performance of the trained OCR is satisfactory.

1.4 Thesis Outline

Chapter 2
In this chapter, we discussed about the previous works that have been done on OCR for Bengali language.
Chapter 3
This chapter contains discussion on different characteristics of Bengali script and important features of digital Bengali fonts that are used in computer.
Chapter 4
This chapter has discussion on different image properties that one must know while developing an OCR.
Chapter 5
We discussed in this chapter about different software that we used in our project.
Chapter 6
In this chapter, we gave an overview of our system and explained why this is different from other projects on Bengali OCR.
Chapter 7
We discussed in detail about the steps of developing the OCR in this chapter.
Chapter 9
How Tesseract works has been explained in this chapter.
Chapter 10
In this chapter, we wrote about our future plan on this OCR and concluded the paper.
Chapter 11
This chapter contains the references.
Chapter 12
This chapter contains the codes that we used to connect Tesseract with the user interface.
REFERENCES
This portion lists the references.
APPENDIX
Appendix contains the codes that we used to connect Tesseract with the user interface.

CHAPTER 2
LITERATURE REVIEW

For Bengali, there are very few OCR solutions as of now. But our government and private organizations have huge quantities of Bengali paper documents that are so important that those should be stored for a long period of time. To do so, making electronic copies of those documents are important and it can be done by using a high-quality Bengali OCR system. But to implement an OCR, the foremost step in the recognition process is the script segmentation of the document image. Since the written form of Bengali documents is more complex than that of many other languages, Bengali script segmentation is of great importance for creating a Bengali OCR system. Though Bengali OCR is not a recent work, but there are very few mentionable works in this field. ‟BOCRA and Apona-Pathak” are two works which were made public in 2006. BOCRA is a recursive acronym that expands to Bocra Optical Character Recognition Application. The final *A* is sort of forced, mainly to make the name pronounceable, but also as an inside joke. The initial *B* could also stand for *Bengali* since that's the primary target language that motivated the authors, but in principle it could be used for other languages as well. In practice, the approach [to be] implemented has no special benefit for languages whose characters/glyphs are separated (i.e. not connected) when printed. But they are not open-sourced. The Center for Research on Bengali Language Processing (CRBLP) released *BengaliOCR*– the first open source OCR software for Bengali – in 2007. *BengaliOCR* is a complete OCR framework, and has a recognition rate of up to 98% but it also has many limitations in its domain.

A team consists of Arif Billah, Al-Mahmud Abdullah and their supervisor Dr. Mumit Khan worked on a survey on Script Segmentation for Bengali OCR. All of them were from BRAC University. According to their work, the primary alphabet of Bengali script is quite large compared to the alphabet sets of English and other western languages. It comprises of 11 vowels, 39 consonants and 10 numerals. The total number of symbols is approximately 300. Besides this huge quantity of symbols, there are various types writing style of those. All these aspects have thrown a great challenge to the researchers in developing a comprehensive OCR for Bengali handwritten scripts.

For both on-line and off-line OCR, recognizing the diversified Bengali handwritten scripts is really tough. Though, some sophisticated research and development has been done on recognition of handwritten Bengali numerals, but very few research works have been found on overall handwritten Bengali OCR .Unlike simple juxtaposition in Roman scripts, each word in Bengali scripts is composed of several characters joined by a horizontal line (called ‘Maatra’or head-line) at the top. Of-ten there may be different composite characters and vowel and consonant signs (‘Kaar’ and ‘Falaa’ symbols).This makes the development of an OCR for Bengali printed scripts a highly challenging task. There are some basic features or properties of any Bengali printed script.

Another work on Bengali OCR was done by S. Mahbub-Uz-Zaman and Tanzina Islam. Former Dean of BRAC University Dr. Mumit khan was their supervisor. They implemented Augmented Reality based text detection and translation application on Android-platform (2.2). This application recognizes the text captured by a mobile phone camera and translates the text and finally displays back the recognized text along with the translation onto the screen. This current version of their application can only translate from Bengali to English. At their first prototype the accuracy rate for Training-set 1 is 68.62 percentages while on the other hand the accuracy rate for the larger training set, Training-set 2 is 84.3 percentages. But the main limitation of them is that there was no detection of spaces and the testing images contained only a few words.

Another team of Md. Abul Hasnat, S.M. Murtoza Habib, and Dr. Mumit Khan worked on this field. They are also from BRAC University .They presented the training and recognition. Their central idea is to separate HMM model for each segmented character or word. They basically emphasized on word level segmentation and like to consider the single character as a word when the character appears alone after segmentation process is done. The system uses HTK toolkit for data preparation, model training from multiple samples and recognition. Features of each trained character are calculated by applying Discrete Cosine Transform (DCT) to each pixel value of the character image where the image is divided into several frames according to its size. The extracted features of each frame are used as discrete probability distributions that will be given as input parameter to each HMM model. In case of recognition a model for each separated character or word is build up using the same approach. This model is given to the HTK toolkit to perform the recognition using Viterbi Decoding. The experimental result shows significant performance.

One professional OCR for Bengali has been developed by a company named ‘Team Engine’. This project was financed by the government of Bangladesh. They demonstrated their OCR on web for a few months. However, currently the demonstration is not available. Its accuracy is said to be around 90%.

CHAPTER 3
IMPORTANT ATTRIBUTES OF BENGALI LANGUAGE

3.1 Characteristics of Bengali Script

Bengali has a very complex script pattern. Not only it has vowels and consonants but also it has combined characters which are composed of several other characters and unclassified characters. Each word in Bengali scripts is formed of a number of characters connected by a horizontal line called 'Maatra' or head-line at the top of the characters but some characters are exception to this. Bengali has a few basic script features. These have been shown below.

a) Style of Bengali writing is from left to right.

b) Bengali does not have the variation of upper and lower case.

c) There are short forms of vowels of Bengali named 'kar' such as ◌া, ◌ি, ◌ৃ and consonants named 'Fola' such as ল্ল, ন্য.

d) In a single syllable of a word, several consonant characters may combine to form a compound character that partly retains the shape of the constituent characters (e.g. Na +Da, Ka + Ta, Va + Ra-falaa, Na + Daa +Rafalaa

e) Except very few characters and symbols (e.g. Ae, Oy, O, Ow, Kha, Ga, Ungo, Nioetc),almost all Bengali alphabets and symbols have a horizontal line at the upper part called 'maatra'. Some are shown in Figure.1a.

f) In a word, the characters with 'maatra' remain connected together through their 'maatra' and other characters and symbols (e.g. Khondota, Bishorgo, Ungo, Ae, Oyetc) remain isolated in the word. They did character, word and line segmentation but didn't confirm the accuracy level of their work.

Table1 shows samples of each type of Bengali characters.

Vowel	**অ আ ই ঈ উ ঊ**
Consonant	**ক খ গ ঘ ঙ চ ছ জ ঝ ঞ ট ঠ ড**
Combined Character	**শ্ব শ্বি শ্ম শ্মা স্ত্র স্বে স্থা স্ফু স্ম স্তী ষ্কা**
Special Character	**র্ষা র্ষী ষাঁ ক্যা ক্যে ক্যো ঝ্যো**

Table1: Samples of characters

3.2 Properties of Digital Bengali Font

It is extremely important to understand how computerized digital Bengali font works while developing an OCR. In Bengali মা is a character that is a combination of ম and the short form of vowel আ. This short form is represented by the short form itself with a circle concatenated with it. Here are few examples of them.

◌া ◌ে ◌ু ◌ী ◌ি

These circles cause difficulty to make the character set of an OCR. It is therefore important for the developers to cover each and every pattern of a character that exists in Bengali language. A certain character ম can have various forms. These are shown below.

ম মা মু মি ম্যা মৌ মূ মি মী

Not only the character can have such forms but also they may be combined to special characters which alone do not have any meaning. One of such special character is 'chandrabindu'. It remains on top of some characters such as 'বাঁ 'to give them special pronunciation. Because of these very complex characteristics of digital Bengali fonts which is actually far different than real life handwritten font, it is very important to cope up with the differences of real life perception about fonts and the digital Bengali fonts that have their completely different way of working.

Chapter 4
IMPORTANT IMAGE PROPERTIES FOR OCR

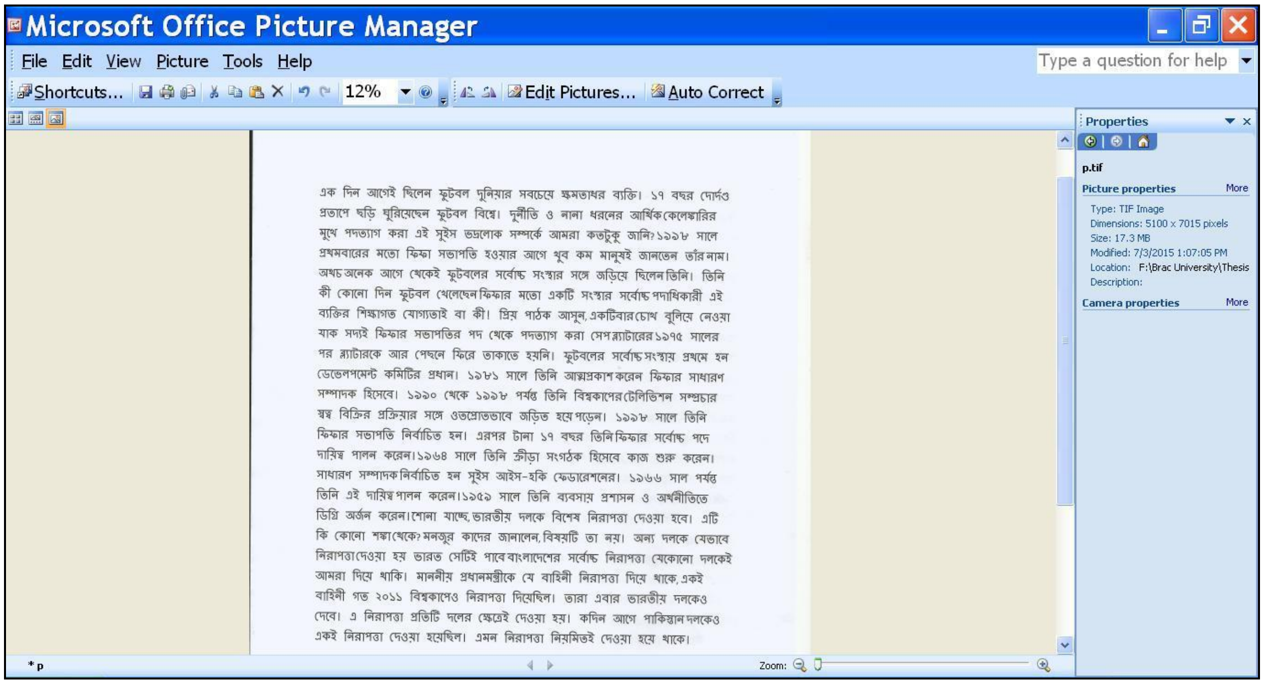
An image is a picture, photograph or any other form of 2D representation of any scene. Since OCR works on images to generate text, development of an OCR is very closely related to different properties that image files have. Some properties are discussed below.

4.1 Noise of Image

Image noise is random (not present in the object imaged) variation of brightness or color information in images, and is usually an aspect of electronic noise. It can be produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that adds spurious and extraneous information. The magnitude of image noise can range from almost imperceptible specks on a digital photograph taken in good light, to optical and radio astronomical images that are almost entirely noise, from which a small amount of information can be derived by sophisticated processing (a noise level that would be totally unacceptable in a photograph since it would be impossible to determine even what the subject was).

High levels of noise are almost always undesirable, but there are cases when a certain amount of noise is useful, for example to prevent discretization artifacts (color banding). Some noise also increases acutance (apparent sharpness). Noise purposely added for such purposes is called dither; it improves the image perceptually, though it degrades the signal-to-noise ratio.

It is very important while developing an OCR to handle the noise of image. In this thesis, we handled this by adding noise to our training images as described later. Figure1 shows an image with noise in the next page.



এক দিন আগেই ছিলেন ফুটবল দুনিয়ার সবচেয়ে ক্ষমতাধর ব্যক্তি। ১৭ বছর দোর্দণ্ড প্রতাপে ছড়ি ঘুরিয়েছেন ফুটবল বিশ্বে। দুর্নীতি ও নানা ধরনের আর্থিক কেলেঙ্কারির মুখে পদত্যাগ করা এই সুইস ভদ্রলোক সম্পর্কে আমরা কতটুকু জানি? ১৯৯৮ সালে প্রথমবারের মতো ফিফা সভাপতি হওয়ার আগে খুব কম মানুষই জানতেন তাঁর নাম। অথচ অনেক আগে থেকেই ফুটবলের সর্বোচ্চ সংস্থার সঙ্গে জড়িয়ে ছিলেন তিনি। তিনি কী কোনো দিন ফুটবল খেলেছেন ফিফার মতো একটি সংস্থার সর্বোচ্চ পদাধিকারী এই ব্যক্তির শিক্ষাগত যোগ্যতাই বা কী। প্রিয় পাঠক আসুন, একটিবার চোখ বুলিয়ে নেওয়া যাক সদ্যই ফিফার সভাপতির পদ থেকে পদত্যাগ করা সেপ ব্ল্যাটারের ১৯৭৫ সালের পর ব্ল্যাটারকে আর পেছনে ফিরে তাকাতে হয়নি। ফুটবলের সর্বোচ্চ সংস্থায় প্রথমে হন ডেভেলপমেন্ট কমিটির প্রধান। ১৯৮১ সালে তিনি আত্মপ্রকাশ করেন ফিফার সাধারণ সম্পাদক হিসেবে। ১৯৯০ থেকে ১৯৯৮ পর্যন্ত তিনি বিশ্বকাপের টেলিভিশন সম্প্রচার স্বত্ব বিক্রির প্রক্রিয়ার সঙ্গে ওতপ্রোতভাবে জড়িত হয়ে পড়েন। ১৯৯৮ সালে তিনি ফিফার সভাপতি নির্বাচিত হন। এরপর টানা ১৭ বছর তিনি ফিফার সর্বোচ্চ পদে দায়িত্ব পালন করেন।১৯৬৪ সালে তিনি ক্রীড়া সংগঠক হিসেবে কাজ শুরু করেন। সাধারণ সম্পাদক নির্বাচিত হন সুইস আইস-হকি ফেডারেশনের। ১৯৬৬ সাল পর্যন্ত তিনি এই দায়িত্ব পালন করেন।১৯৫৯ সালে তিনি ব্যবসায় প্রশাসন ও অর্থনীতিতে ডিগ্রি অর্জন করেন।শোনা যাচ্ছে, ভারতীয় দলকে বিশেষ নিরাপত্তা দেওয়া হবে। এটি কি কোনো শঙ্কা থেকে? মনজুর কাদের জানালেন, বিষয়টি তা নয়। অন্য দলকে যেভাবে নিরাপত্তা দেওয়া হয় ভারত সেটিই পাবে বাংলাদেশের সর্বোচ্চ নিরাপত্তা যেকোনো দলকেই আমরা দিয়ে থাকি। মাননীয় প্রধানমন্ত্রীকে যে বাহিনী নিরাপত্তা দিয়ে থাকে, একই বাহিনী গত ২০১১ বিশ্বকাপেও নিরাপত্তা দিয়েছিল। তারা এবার ভারতীয় দলকেও দেবে। এ নিরাপত্তা প্রতিটি দলের ক্ষেত্রেই দেওয়া হয়। কদিন আগে পাকিস্তান দলকেও একই নিরাপত্তা দেওয়া হয়েছিল। এমন নিরাপত্তা নিয়মিতই দেওয়া হয়ে থাকে।

p.tif
Type: TIF Image
Dimensions: 5100 x 7015 pixels
Size: 17.3 MB
Modified: 7/3/2015 1:07:05 PM
Location: F:\Brac University\Thesis
Description:

Figure1: Image with noise

4.2 Skewness of Image

Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution.

In a normal distribution, the graph appears as a classical, symmetrical "bell-shaped curve." The mean, or average, and the mode, or maximum point on the curve, are equal.

- In a perfect normal distribution, the tails on either side of the curve are exact mirror images of each other.
- When a distribution is skewed to the left, the tail on the curve's left-hand side is longer than the tail on the right-hand side, and the mean is less than the mode. This situation is also called negative skewness.
- When a distribution is skewed to the right the tail on the curve's right-hand side is longer than the tail on the left-hand side, and the mean is greater than the mode. This situation is also called positive skewness.

Figure2 shows an example of skewness curve.

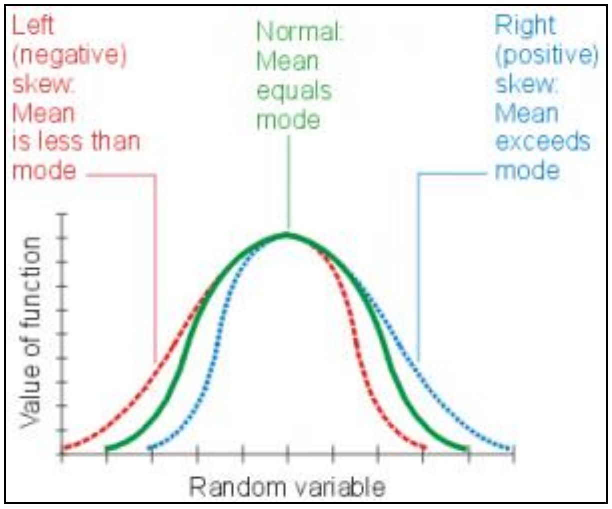


Figure2: Skewness graph

So, an image can be either positively skewed or negatively skewed. In this research, we did not add any special measure for handling skewness. Tesseract can handle limited 2D skewness without any particular training form with the help of its own algorithm. Figure1 is also an example of skewness.

4.3 Formats of Image

Image file formats are standardized means of organizing and storing digital images. Image files are composed of digital data in one of these formats that can be rasterized for use on a computer display or printer. An image file format may store data in uncompressed, compressed, or vector formats. Once rasterized, an image becomes a grid of pixels, each of which has a number of bits to designate its color equal to the color depth of the device displaying it.

Including proprietary types, there are hundreds of image file types. The PNG, JPEG, and GIF formats are most often used to display images on the Internet. However, to train Tesseract, one must use TIF or Tagged Image File Format. It is an image format that normally saves eight bits or sixteen bits per color (red, green, blue) for 24-bit and 48-bit totals, respectively, usually using either the TIFF or TIF filename extension. The reason why Tesseract needs this format is that Optical Character Recognition software packages commonly generate some form of TIFF image (often monochromatic) for scanned text pages. However, it can take TIF and PNG formatted images as input while PNG is the preferred format.

Chapter 5
BACKGROUND OF OCR DEVELOPING COMPONENTS

5.1 Tesseract OCR Engine

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006.

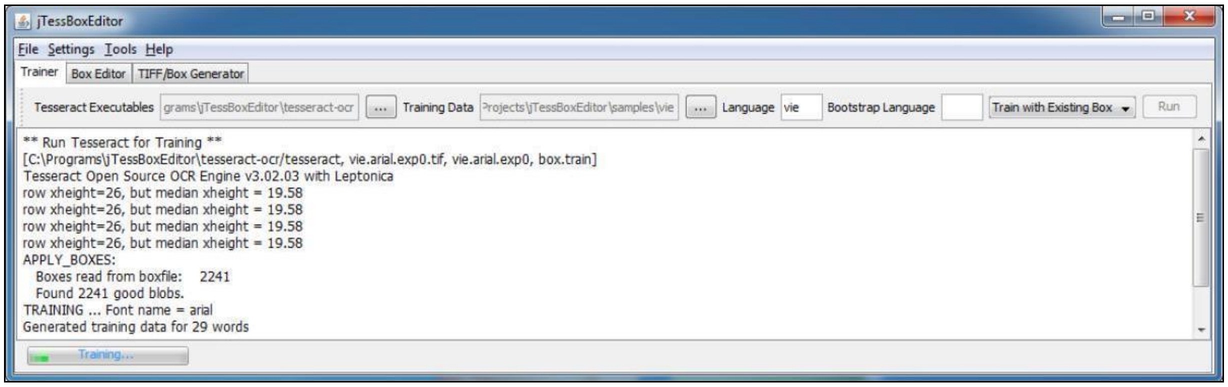
The Tesseract engine was originally developed as proprietary software at Hewlett Packard labs in Bristol, England and Greeley, Colorado between 1985 and 1994, with some more changes made in 1996 to port to Windows, and some migration from C to C++ in 1998. A lot of the code was written in C and then some more was written in C++. Since then all the code has been converted to at least compile with a C++ compiler.[1] It was then released as open source in 2005 by Hewlett Packard and the University of Nevada, Las Vegas (UNLV).

Tesseract is available for Linux, Windows and Mac OS X, however, due to limited resources only Windows and Ubuntu are rigorously tested by developers.

Tesseract up to and including version 2 could only accept TIFF images of simple one column text as inputs. These early versions did not include layout analysis and so inputting multi-columned text, images, or equations produced a garbled output. Since version 3.00 Tesseract has supported output text formatting, hOCR positional information and page layout analysis. Support for a number of new image formats was added using the Leptonica library. Tesseract is suitable for use as a backend. Tesseract does not come with a GUI and is instead run from the command-line interface.

5.2 JTessBoxEditor

JTessBoxEditor is a box editor and trainer for Tesseract OCR, providing editing of box data of both Tesseract 2.0x and 3.0x formats. It can read common image formats, including multi-page TIFF. The program requires Java Runtime Environment 7 or later. JTessBoxEditor is released and distributed under the Apache License, v2.0. Figure3 shows the JTessboxEditor in the next page.



Tesseract Executables: grams\jTessBoxEditor\tesseract-ocr
Training Data: Projects\jTessBoxEditor\samples\vie
Language: vie
Bootstrap Language:

```
** Run Tesseract for Training **
[C:\Programs\jTessBoxEditor\tesseract-ocr/tesseract, vie.arial.exp0.tif, vie.arial.exp0, box.train]
Tesseract Open Source OCR Engine v3.02.03 with Leptonica
row xheight=26, but median xheight = 19.58
row xheight=26, but median xheight = 19.58
row xheight=26, but median xheight = 19.58
row xheight=26, but median xheight = 19.58
APPLY_BOXES:
  Boxes read from boxfile:    2241
  Found 2241 good blobs.
TRAINING ... Font name = arial
Generated training data for 29 words
```

Figure3: JTessBoxEditor

5.3 NetBeans IDE

NetBeans is a software development platform written in Java. The NetBeans Platform allows applications to be developed from a set of modular software components. Applications based on the NetBeans Platform, including the NetBeans integrated development environment (IDE), can be extended by third party developers. NetBeans is cross-platform and runs on Microsoft Windows, Mac OS X, Linux, Solaris and other platforms supporting a compatible JVM.

NetBeans began in 1996 as Xelfi, a Java IDE student project under the guidance of the Faculty of Mathematics and Physics at Charles University in Prague. Net beans has been bought by Sun Microsystems in 1999. Sun open-sourced the NetBeans IDE in June of the following year. Since then, the NetBeans community has continued to grow. In 2010, Sun (and thus NetBeans) was acquired by Oracle.

CHAPTER 6
SYSTEM OVERVIEW

6.1 Distinctive Features of Our System

This project is an example of training a software engine for language processing.

1. This project has been implemented in a comprehensive manner, thus, it covers a lot of aspects of Bengali language.
2. Testing files used in this project are real life standard.
3. This project is very user friendly to use.
4. Developing the system in Windows Operating System makes it more accessible to the users.
5. Its maximum accuracy is better compared to previous projects accomplished.
6. It is very light software although its function is very intricate.
7. It is portable, so no installation is required.

6.2 Overview of Developing Procedure

The system is based on Tesseract OCR Engine with the user interface developed in Java Graphical User Interface platform. The OCR Engine needs a library file to work on called 'traineddata'. This file is composed of several other files. It is a concatenation of those files. When the library file is put into a specific location, Tesseract can access it for its scanning and text generating purpose. Tesseract has a level of accuracy in its engine which is standard. This engine can work to its full potential provided the library file is accurate and rich. In our system, we have implemented the library file or Traineddata in a very detailed manner. We have covered all sorts of Bengali letters for the 'Solaimanlipi' font. Bengali, as a language is quite complicated and it has got various types of letters including vowel, consonants, combined characters and other anonymous types. We developed a huge character set of all types after a long research. It is important to mention that we needed to uniquely identify each and every character in our system so that if the input file contains the character, the OCR recognizes it.

We needed to make all letters that are covered in the newspapers, books, journals, etc. We prepared 'tif' formatted image files of them. Then we used those images to make the training files of our traineddata. Because of the complexity of the character set, the OCR may not always detect a character correctly even if the character is included in training files. Tesseract can be manipulated in both Windows and Linux. As Windows is relatively popular operating system, we preferred to work on it. We installed the executable file of Tesseract in windows. In the command line of windows, we executed several commands to prepare our training files from the images. Then we concatenated those files using commands to make the traineddata.

We used Netbeans IDE to make the user interface. We developed the interface in JAVA. The entire system is formed as a zip package. The zip contains the interface, OCR engine and the traineddata. The input can be selected using the interface. Output will be saved in the project's folder. Then the user can use the OCR through the interface. We show the sequence of system development in Figure4 in the next page.

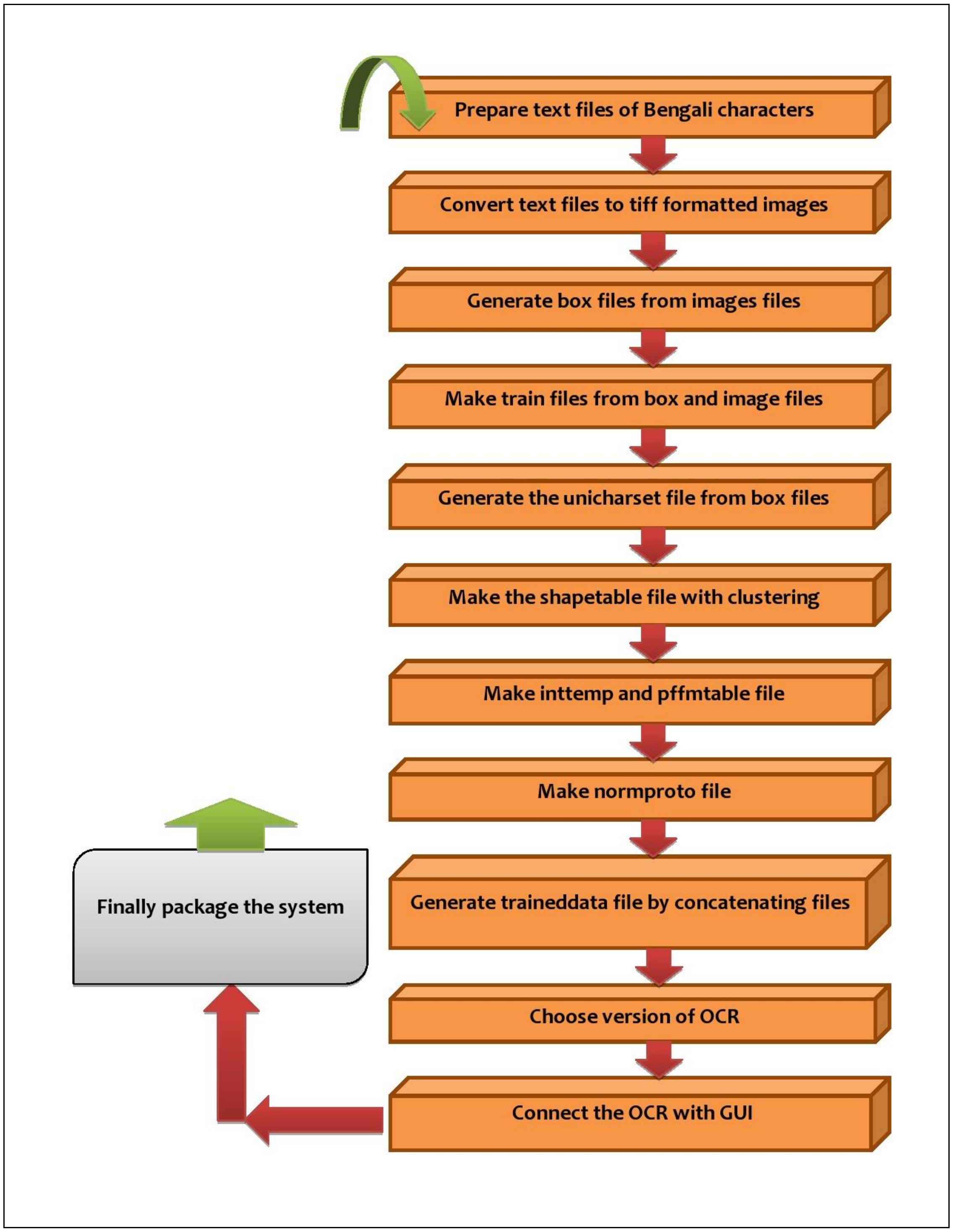

Prepare text files of Bengali characters
Convert text files to tiff formatted images
Generate box files from images files
Make train files from box and image files
Generate the unicharset file from box files
Make the shapetable file with clustering
Make inttemp and pffmtable file
Make normproto file
Generate traineddata file by concatenating files
Choose version of OCR
Connect the OCR with GUI
Finally package the system

Figure4: Block Diagram of System Development

Figure5 shows the step of use.

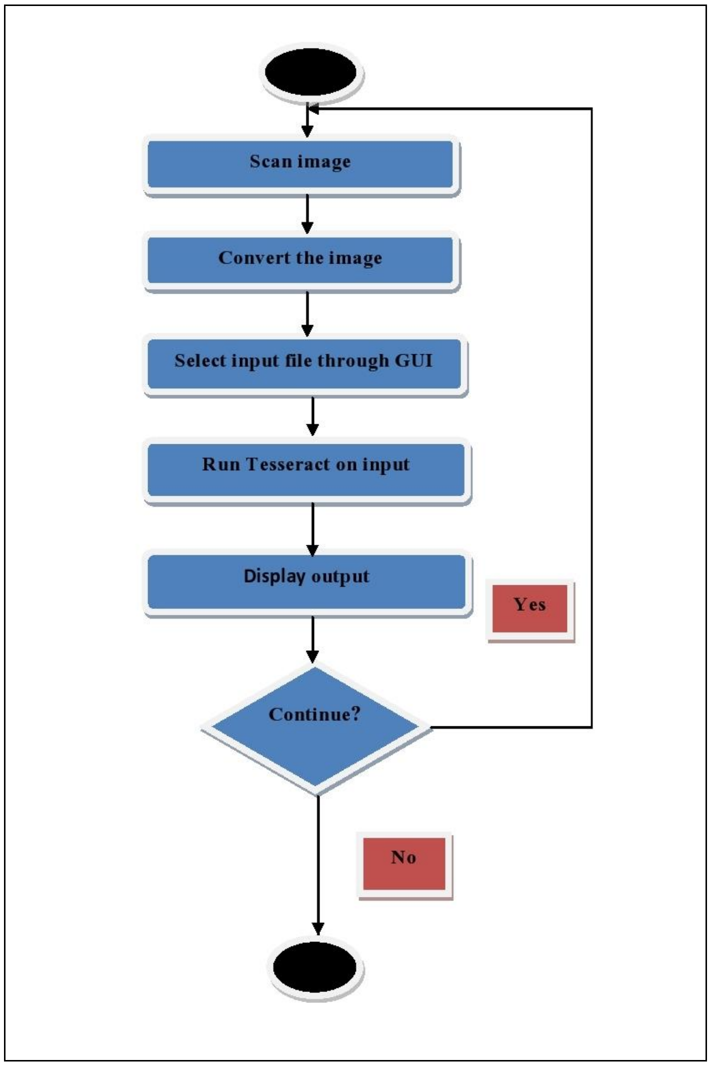


Figure5: Steps of use

The accuracy of the output will vary according to the quality of input images. If the input file is a photo taken by a camera, it is important to notice if there is any shadow. If there is shadow, it will cause trouble for the OCR to differentiate between the shadow and the text. If the input file is obtained by scanning a document, it is likely to produce better quality outputs. The resolution of the scanned file should not be less than 400 pixels.

CHAPTER 7
SYSTEM DEVELOPMENT DESCRIPTION

7.1 Installing the OCR Engine

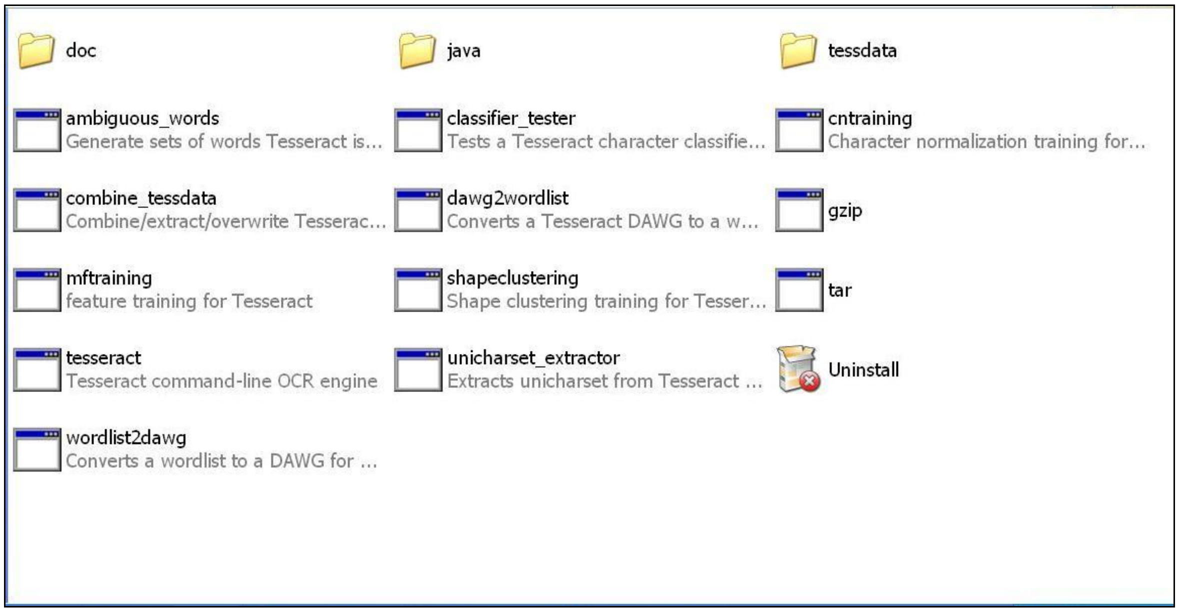
We installed Tesseract in Windows Operating System. Google provides an installer of Tesseract with built-in English traindata. We installed the installer. A folder named 'Tesseract-OCR' was created in the 'Program Files' folder. There is no user interface for Tesseract. So, we needed to use the command line to run it. We used the following commands:

cd C:\

cd Program Files

cd Tesseract-OCR

After the following commands, we could access the Tesseract-OCR and its different components. Figure6 shows a view of the Tesseract-OCR folder.

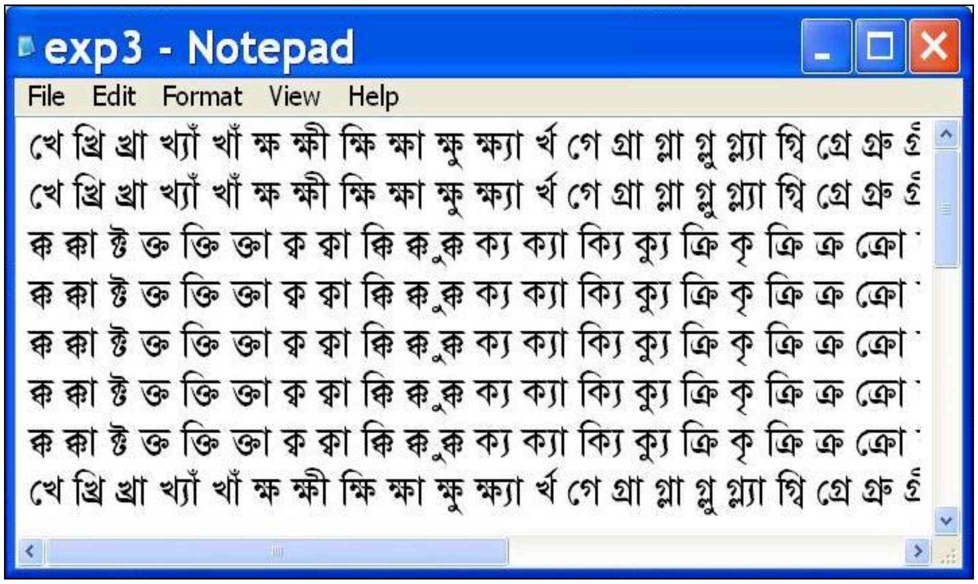


doc
java
tessdata
ambiguous_words
Generate sets of words Tesseract is...
classifier_tester
Tests a Tesseract character classifie...
cntraining
Character normalization training for...
combine_tessdata
Combine/extract/overwrite Tesserac...
dawg2wordlist
Converts a Tesseract DAWG to a w...
gzip
mftraining
feature training for Tesseract
shapeclustering
Shape clustering training for Tesser...
tar
tesseract
Tesseract command-line OCR engine
unicharset_extractor
Extracts unicharset from Tesseract ...
Uninstall
wordlist2dawg
Converts a wordlist to a DAWG for ...

Figure6: Tesseract Installation folder

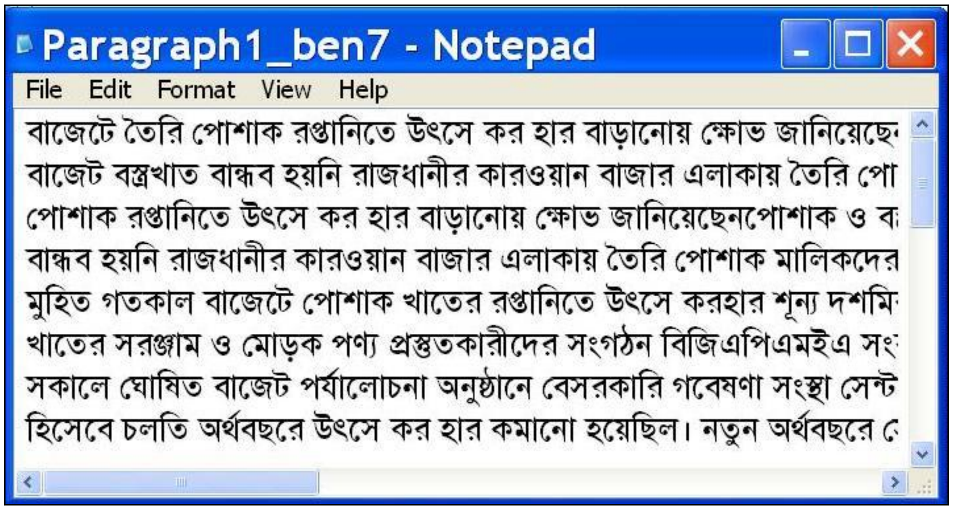
7.2 Preparing text Files

Tesseract takes tiff formatted images files as inputs to make the traindata. The images files can be a scanned image of a printed paper, an electronically converted image from a text file or an image taken by a camera. The usual practice is to convert text files to prepare images. We prepared eleven text files. It is important to note that the encoding of the text files needs to be UTF-8 otherwise the converter will not be able to read the Bengali text. The first six text files contain the Bengali vowels, consonants, combined characters and unclassified characters. The last five text files contain paragraphs. In the first six files, each character is separated by a space. Figure7 (a) shows a text file with characters and Figure7 (b) shows a text file with paragraph.



খে খ্রি খ্রা খ্যাঁ খাঁ ক্ষ ক্ষী ক্ষি ক্ষা ক্ষু ক্ষ্যা র্খ গে গ্রা গ্লা গ্লু গ্ল্যা গ্বি গ্রে গ্রু
খে খ্রি খ্রা খ্যাঁ খাঁ ক্ষ ক্ষী ক্ষি ক্ষা ক্ষু ক্ষ্যা র্খ গে গ্রা গ্লা গ্লু গ্ল্যা গ্বি গ্রে গ্রু
ক্ক ক্কা ক্ট ক্ত ক্তি ক্তা ক্ব ক্বা ক্কি ক্ক্ক ক্য ক্যা ক্যি ক্যু ক্রি কৃ ক্রি ক্র ক্রো
ক্ক ক্কা ক্ট ক্ত ক্তি ক্তা ক্ব ক্বা ক্কি ক্ক্ক ক্য ক্যা ক্যি ক্যু ক্রি কৃ ক্রি ক্র ক্রো
ক্ক ক্কা ক্ট ক্ত ক্তি ক্তা ক্ব ক্বা ক্কি ক্ক্ক ক্য ক্যা ক্যি ক্যু ক্রি কৃ ক্রি ক্র ক্রো
ক্ক ক্কা ক্ট ক্ত ক্তি ক্তা ক্ব ক্বা ক্কি ক্ক্ক ক্য ক্যা ক্যি ক্যু ক্রি কৃ ক্রি ক্র ক্রো
ক্ক ক্কা ক্ট ক্ত ক্তি ক্তা ক্ব ক্বা ক্কি ক্ক্ক ক্য ক্যা ক্যি ক্যু ক্রি কৃ ক্রি ক্র ক্রো
খে খ্রি খ্রা খ্যাঁ খাঁ ক্ষ ক্ষী ক্ষি ক্ষা ক্ষু ক্ষ্যা র্খ গে গ্রা গ্লা গ্লু গ্ল্যা গ্বি গ্রে গ্রু

Figure7 (a): Sample text file with characters for training



বাজেটে তৈরি পোশাক রপ্তানিতে উৎসে কর হার বাড়ানোয় ক্ষোভ জানিয়েছে
বাজেট বস্ত্রখাত বান্ধব হয়নি রাজধানীর কারওয়ান বাজার এলাকায় তৈরি পো
পোশাক রপ্তানিতে উৎসে কর হার বাড়ানোয় ক্ষোভ জানিয়েছেনপোশাক ও ব
বান্ধব হয়নি রাজধানীর কারওয়ান বাজার এলাকায় তৈরি পোশাক মালিকদের
মুহিত গতকাল বাজেটে পোশাক খাতের রপ্তানিতে উৎসে করহার শূন্য দশমি
খাতের সরঞ্জাম ও মোড়ক পণ্য প্রস্তুতকারীদের সংগঠন বিজিএপিএমইএ সং
সকালে ঘোষিত বাজেট পর্যালোচনা অনুষ্ঠানে বেসরকারি গবেষণা সংস্থা সেন্ট
হিসেবে চলতি অর্থবছরে উৎসে কর হার কমানো হয়েছিল। নতুন অর্থবছরে

Figure7 (b): Sample text file with paragraph for training

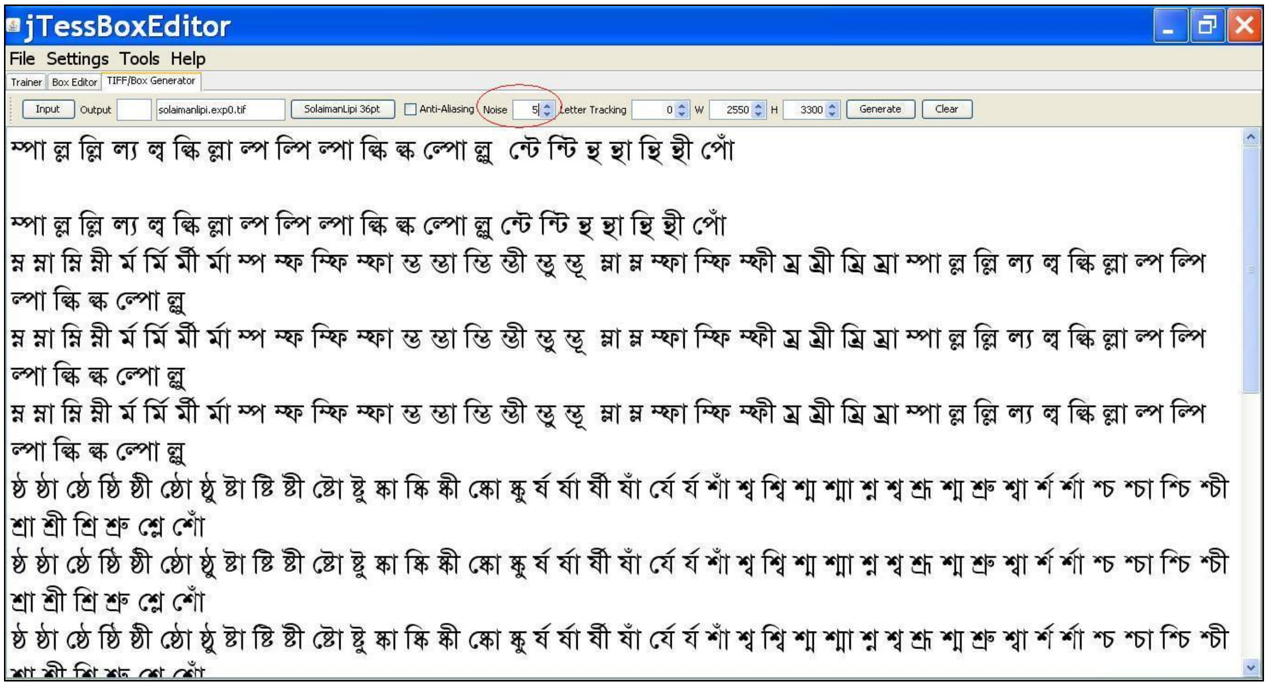
7.3 Making Image Files with Noise Margin

We converted text files to images through a converter. There is a software named 'Jtessbox Editor' for Tesseract. This software is based on java and it requires the java runtime environment (JRE).

We converted the text files to tiff images using the Editor. Past research suggests that it is good to add some artificial noise to the training images which helps the OCR to work better on scanned image files. We set the noise margin's value as 5 in Jtessbox Editor. The Editor generated a text file called box file for each tiff image. Box file is a UTF-8 encoded text file that has the coordinates of all the characters in the training images along with the characters. Each character has a certain abstract edge and the edge is called a box. Each box has top, bottom, left, right coordinate value. For instance a particular box will have coordinates like the following example

ক 100 200 350 400

Figure8 shows an example of text to image conversion with the noise margin highlighted by a red circle.



solaimanlipi.exp0.tif | SolaimanLipi 36pt | Anti-Aliasing ☐ | Noise 5 | Letter Tracking 0 | W 2550 | H 3300

স্পা ল্ল ল্লি ল্য ল্ব ল্কি ল্লা ল্প ল্পি ল্পা ল্কি ল্ক ল্পো ল্লু ন্টে ন্টি হ্ব হ্বা হ্বি হ্বী পোঁ

স্পা ল্ল ল্লি ল্য ল্ব ল্কি ল্লা ল্প ল্পি ল্পা ল্কি ল্ক ল্পো ল্লু ন্টে ন্টি হ্ব হ্বা হ্বি হ্বী পোঁ
ম্ন ম্না ম্নি ম্নী র্ম র্মি র্মী র্মা স্প স্ফ স্ফি স্ফা স্ত স্তা স্তি স্তী স্তু স্তূ ম্লা ম্ল স্ফা স্ফি স্ফী ম্র ম্রী ম্রি ম্রা স্পা ল্ল ল্লি ল্য ল্ব ল্কি ল্লা ল্প ল্পি
ল্পা ল্কি ল্ক ল্পো ল্লু
ম্ন ম্না ম্নি ম্নী র্ম র্মি র্মী র্মা স্প স্ফ স্ফি স্ফা স্ত স্তা স্তি স্তী স্তু স্তূ ম্লা ম্ল স্ফা স্ফি স্ফী ম্র ম্রী ম্রি ম্রা স্পা ল্ল ল্লি ল্য ল্ব ল্কি ল্লা ল্প ল্পি
ল্পা ল্কি ল্ক ল্পো ল্লু
ম্ন ম্না ম্নি ম্নী র্ম র্মি র্মী র্মা স্প স্ফ স্ফি স্ফা স্ত স্তা স্তি স্তী স্তু স্তূ ম্লা ম্ল স্ফা স্ফি স্ফী ম্র ম্রী ম্রি ম্রা স্পা ল্ল ল্লি ল্য ল্ব ল্কি ল্লা ল্প ল্পি
ল্পা ল্কি ল্ক ল্পো ল্লু
ষ্ঠ ষ্ঠা ষ্ঠে ষ্ঠি ষ্ঠী ষ্ঠো ষ্ঠু ষ্টা ষ্টি ষ্টী ষ্টো ষ্টু ষ্কা ষ্কি ষ্কী ষ্কো ষ্কু র্য র্যা র্যী র্যাঁ র্যে র্য শাঁ শ্ব শ্বি শ্ম শ্মা শ্ন শ্ব শ্রূ শ্ম শ্রু শ্বা র্শ র্শা শ্চ শ্চা শ্চি শ্চী
শ্রা শ্রী শ্রি শ্রু শ্লে শোঁ
ষ্ঠ ষ্ঠা ষ্ঠে ষ্ঠি ষ্ঠী ষ্ঠো ষ্ঠু ষ্টা ষ্টি ষ্টী ষ্টো ষ্টু ষ্কা ষ্কি ষ্কী ষ্কো ষ্কু র্য র্যা র্যী র্যাঁ র্যে র্য শাঁ শ্ব শ্বি শ্ম শ্মা শ্ন শ্ব শ্রূ শ্ম শ্রু শ্বা র্শ র্শা শ্চ শ্চা শ্চি শ্চী
শ্রা শ্রী শ্রি শ্রু শ্লে শোঁ
ষ্ঠ ষ্ঠা ষ্ঠে ষ্ঠি ষ্ঠী ষ্ঠো ষ্ঠু ষ্টা ষ্টি ষ্টী ষ্টো ষ্টু ষ্কা ষ্কি ষ্কী ষ্কো ষ্কু র্য র্যা র্যী র্যাঁ র্যে র্য শাঁ শ্ব শ্বি শ্ম শ্মা শ্ন শ্ব শ্রূ শ্ম শ্রু শ্বা র্শ র্শা শ্চ শ্চা শ্চি শ্চী
শ্রা শ্রী শ্রি শ্রু শ্লে শোঁ

Figure8: Conversion of text to image

7.4 Editing Box Files

In Bengali fonts, short forms of vowels such as ◌া ◌ু ◌ি ◌ী are treated as separate characters. So, JtessboxEditor which has Tesseract's training engine inside it split the short form of the vowels. So, we needed to merge them to make characters like মা (ম+◌া) হা (হ+আ) সু

(স+ু), শি (শ+ি).The editor has the options. We used them to get the requisite formatted characters such as মা, বাবা, চাচা. However, for special characters like ং and ঃ, merging the box files needed special measures. We needed to manually edit the images with photo editor and then generate box files and edit them.

Figure9 shows the state of box before and after merging in the next page.

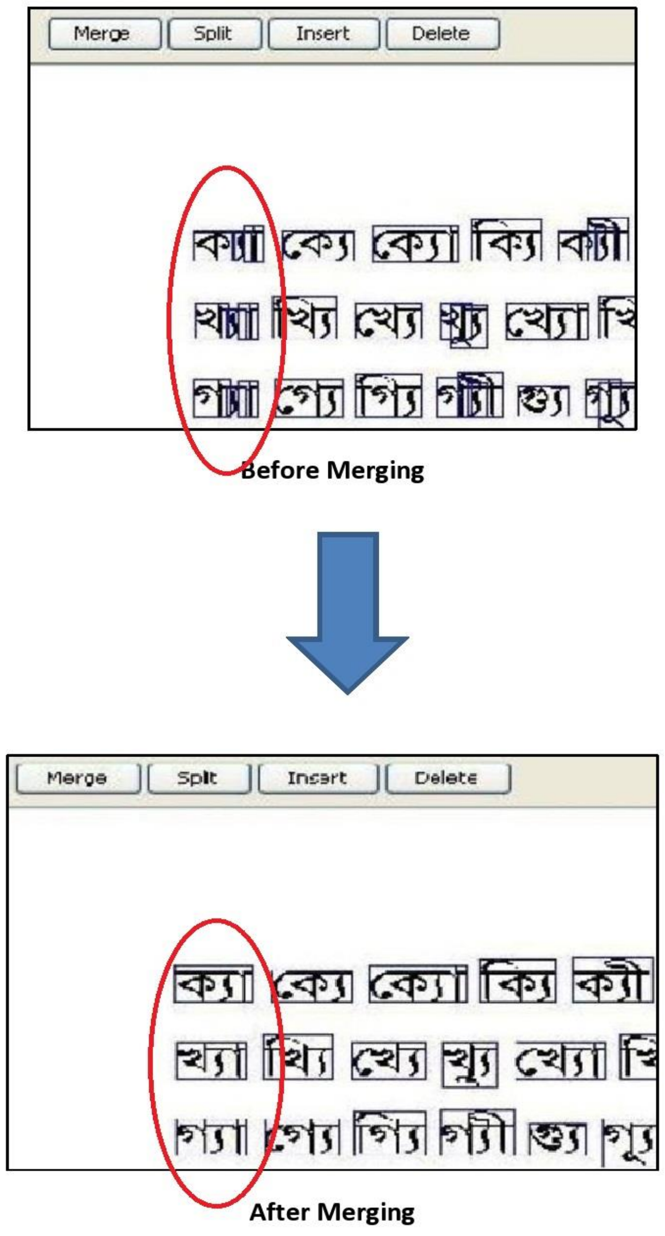
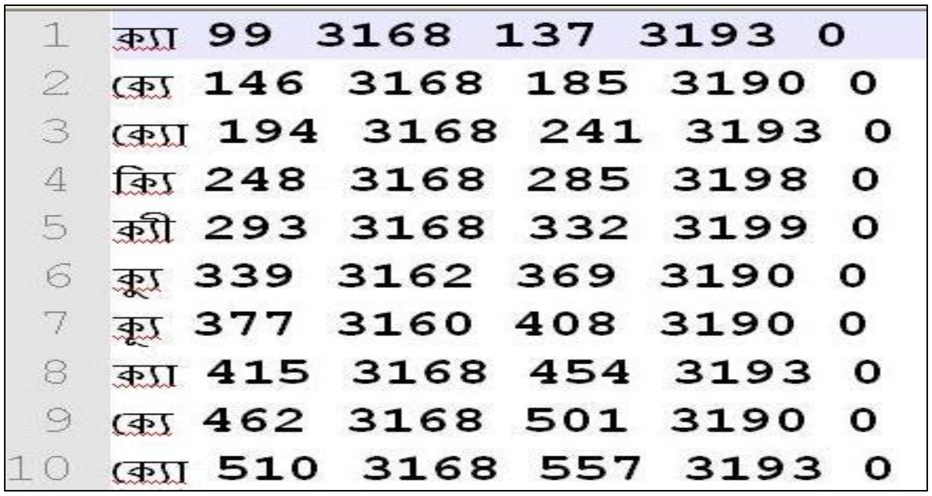


Figure9: Merging steps of box file

Figure10 shows one such box file in the editor with coordinates.



```
ক্যা 99  3168 137 3193 0
ক্যে 146 3168 185 3190 0
ক্যো 194 3168 241 3193 0
ক্যি 248 3168 285 3198 0
ক্যী 293 3168 332 3199 0
ক্যু 339 3162 369 3190 0
ক্যূ 377 3160 408 3190 0
ক্যা 415 3168 454 3193 0
ক্যে 462 3168 501 3190 0
ক্যো 510 3168 557 3193 0
```

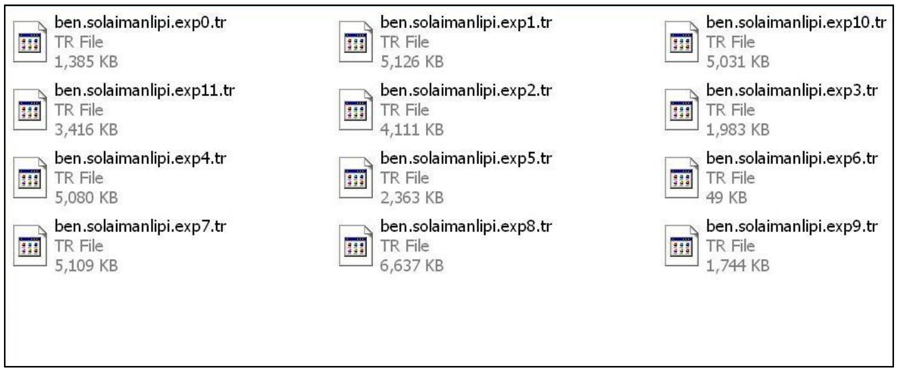
Figure10: Box file with coordinates

7.5 Running Tesseract for Training

For each pair of tiff image and box file, we ran Tesseract for training. It generated a file with extension 'tr' for each pair of tiff image and box file. These files contain the training information. Tesseract scans a character from the input, tries to match it with a shape available in the tiff image which is in the traindata and write the matching character from the box file in the output. So, we needed to cover all sorts of available shapes in the training images. We used the following command for creating each tr file.

tesseract ben.solaimanlipi.exp0.tif ben.solaimanlpi.exp0.box.train

Here, ben is the International Organization for Standardization's (ISO) selected prefix for Bengali that we needed to use. Figure11 gives a view of generated training files.



```
ben.solaimanlipi.exp0.tr     ben.solaimanlipi.exp1.tr     ben.solaimanlipi.exp10.tr
TR File                      TR File                      TR File
1,385 KB                     5,126 KB                     5,031 KB
ben.solaimanlipi.exp11.tr    ben.solaimanlipi.exp2.tr     ben.solaimanlipi.exp3.tr
TR File                      TR File                      TR File
3,416 KB                     4,111 KB                     1,983 KB
ben.solaimanlipi.exp4.tr     ben.solaimanlipi.exp5.tr     ben.solaimanlipi.exp6.tr
TR File                      TR File                      TR File
5,080 KB                     2,363 KB                     49 KB
ben.solaimanlipi.exp7.tr     ben.solaimanlipi.exp8.tr     ben.solaimanlipi.exp9.tr
TR File                      TR File                      TR File
5,109 KB                     6,637 KB                     1,744 KB
```

Figure11: Train files

7.6 Generating the Unicharset File

Tesseract needs to know the set of possible characters it can generate as output. For this reason, it needs a file named unicharset. To generate the unicharset data file, we used the unicharset_extractor program on the box files generated above:

unicharset_extractor ben.solaimanlipi.exp0.box ben.solaimanlipi.exp1.box ..

Figure12 shows the unicharset file.

Figure12: The unicharset file

7.7 Setting Font Properties

The latest version of Tesseract needs us to set the font's properties for our language. Font properties indicate to the different properties a particular font can assume. For instance, an English font can have the properties of bold and italic. However, since Bengali can only be bold, it has only one property. In Tesseract, it is set by a text file that contains a line for each font. Since we worked on a single font, our file has only one line. The line is as follows:

solaimanlipi 0 1 0 0 0

This line refers to the following line

<fontname><italic><bold><fixed><serif><fraktur>

We made the position of bold to 1 to make it set while we reset everything else. The font properties file is a UTF-8 encoded text file.

7.8 Clustering

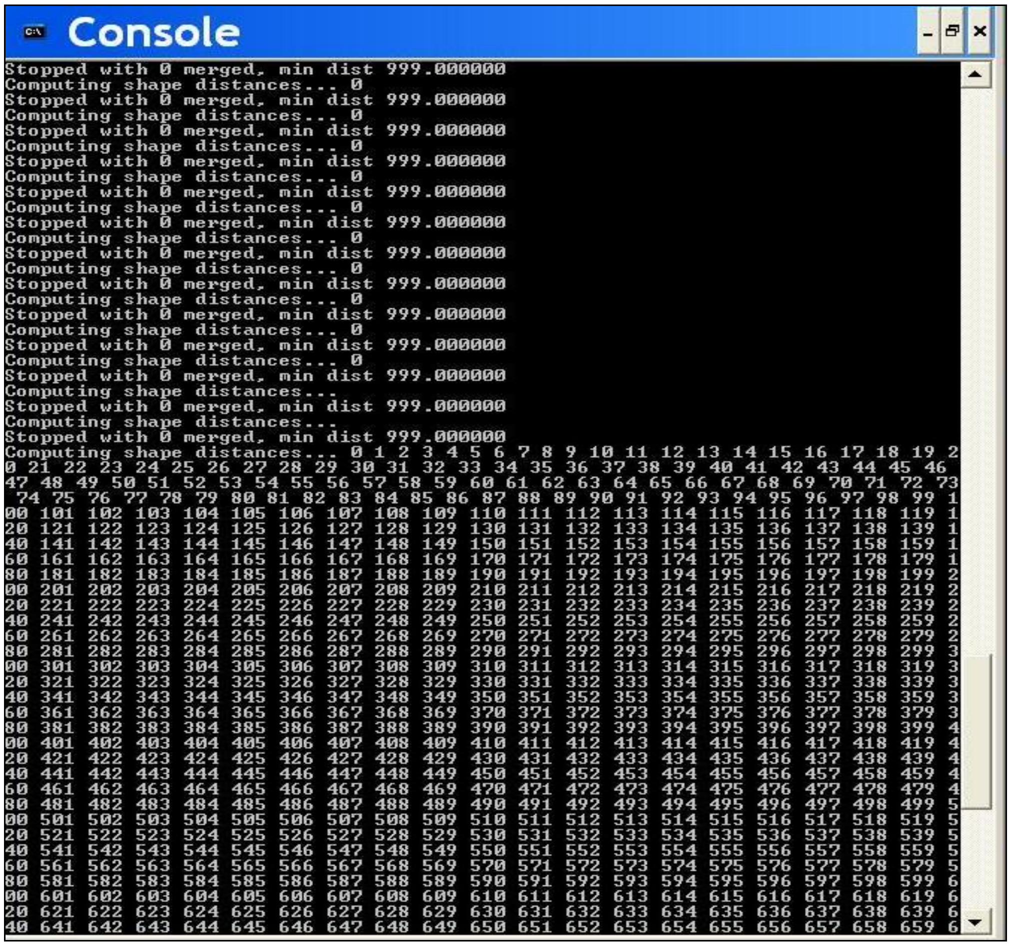
When the alphabetical features of all the training images have been extracted, we needed to cluster them to generate the prototypes. The character shape features can be clustered using the shapeclustering, mftraining and cntraining programs. We have given below the commands to run these programs.

shapeclustering -F font_properties.txt -U unicharset ben.solaimanlipi.exp0.tr ben.solaimanlipi.exp1.tr...

This creates a file named shapetable. Next, we needed to execute another command

mftraining -F font_properties.txt -U unicharset -O ben.unicharsetben.solaimanlipi.exp0.tr ben.solaimanlipi.exp1.tr..

Figure13 shows the command line execution for shapeclustering in the next page.



```
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0
Stopped with 0 merged, min dist 999.000000
Computing shape distances...
Stopped with 0 merged, min dist 999.000000
Computing shape distances...
Stopped with 0 merged, min dist 999.000000
Computing shape distances... 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 2
0 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 1
00 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 1
20 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 1
40 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 1
60 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 1
80 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 2
00 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 2
20 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 2
40 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 2
60 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 2
80 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 3
00 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 3
20 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 3
40 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 3
60 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 3
80 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 4
00 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 4
20 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 4
40 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 4
60 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 4
80 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 5
00 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 5
20 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 5
40 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 5
60 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 5
80 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 6
00 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 6
20 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 6
40 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 6
```

Figure13: Command Line Execution

This command created two files named inttemp and pffmtable. The inttemp file contains the key information of traindata. The last command of clustering is cntraining:

cntraining ben.solaimanlipi.exp0.tr ben.solaimanlipi.exp1.tr…

This produced the normproto data file (The character normalization sensitivity prototypes).

Figure14 shows the inttemp, shapetable, pffmtable and normproto data file.

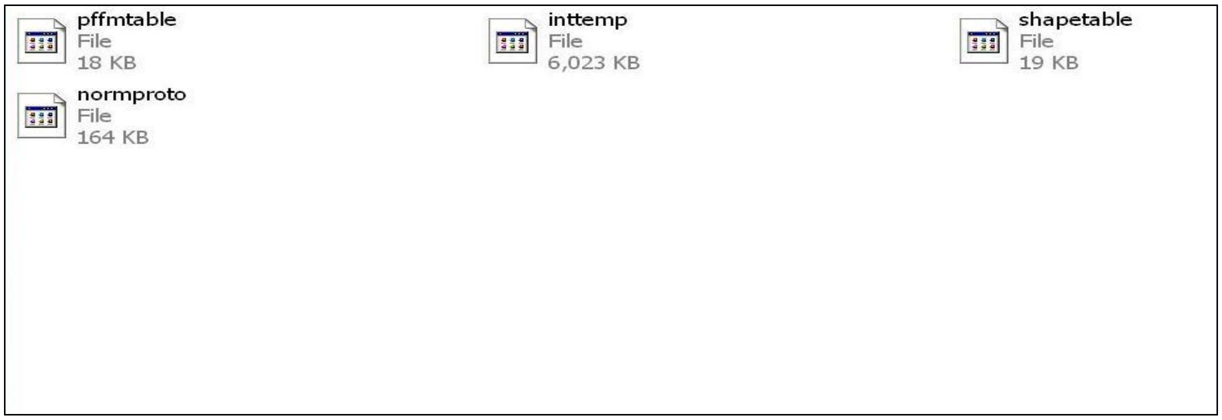


Figure14: shapetable, inttemp, pffmtable and normproto file

7.9 Making Optional Dictionary Data

Tesseract uses up to eight dictionary files for each language. These are optional and assist Tesseract to decide the probability of occurrence of different possible character combinations. We made two dictionary data files. They are wordlist and frequent wordlist. The wordlist file contains around 2000 general random Bengali words and the frequent wordlist contains around 600 common Bengali words such as মা, বাবা, খেলা, আকাশ, etc.

We prepared a text file for each type of list in UTF-8 encoded form. In each file, a word is written in every line. It looks like the following lines:

আকাশ

বাতাস

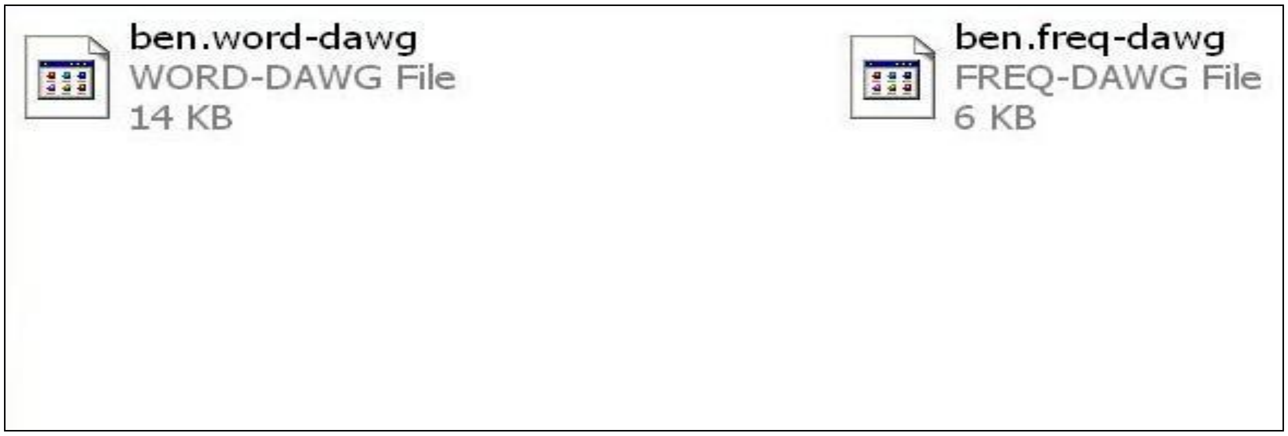
মাটি

Then we ran the following commands.

wordlist2dawg words_list.txt ben.word-dawg ben.unicharset

wordlist2dawg frequent_words_list.txt ben.freq-dawgben.unicharset

They created two dawg(Directed Acyclic Word Graph) files. Figure15 shows the dictionary data files in the next page.



ben.word-dawg
WORD-DAWG File
14 KB

ben.freq-dawg
FREQ-DAWG File
6 KB

Figure15: Dawg files

7.10 Generating Traineddata

After all these files have been prepared, we had to rename the shapetable, pffmtable, normproto and inttemp files according to our language code. For Bengali, we have used language = ‘ben’ as mentioned before. After renaming, the above four files will be named like the following.
shapetable -> ben. shapetable

pffmtable -> ben.pffmtable

normproto -> ben.normproto

inttemp -> ben.inttemp

Then we ran the following command to concatenate them to generate our traindata.

combine tessdata ben. (Here ‘.’ is essential)

The traineddata is now our library for working with Bengali letters. Tesseract’s OCR engine uses its algorithm on this traineddata to produce Bengali outputs on inputs. The traineddata contains 18110 characters and 2617 words. Figure16 shows the concatenated traineddata.

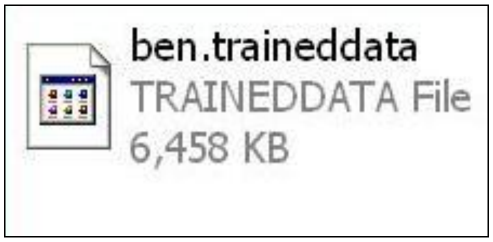

ben.traineddata
TRAINEDDATA File
6,458 KB

Figure16: The traineddata

7.11 Developing Graphical User Interface

We developed a graphical user interface as there is no user interface in Tesseract. This interface was implemented in Netbeans IDE using the built-in exec method in java. Figure17 shows the interface.

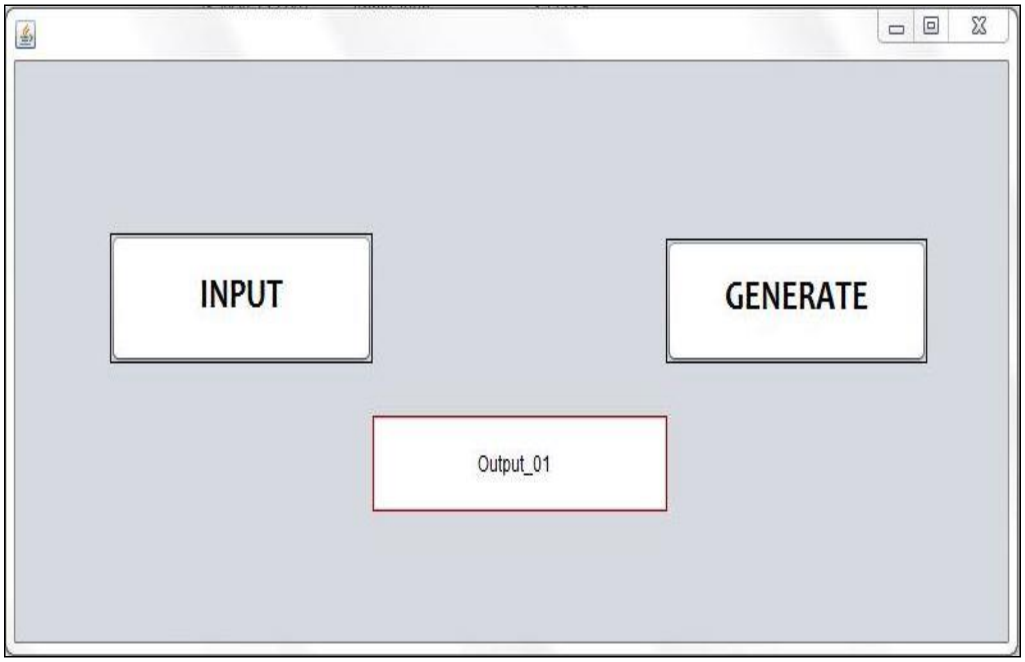


Figure17: User Interface

7.12 Managing Versions

Version 3.03 of Tesseract OCR is used for generating outputs from Bengali text files. It is a beta version so it could not be used for training the OCR engine and we used the version3.02 instead for making the library or traineddata for the OCR. Version 3.03 has been integrated to the system and has given better performance than that of Version 3.02. Though the entire training procedure is based on version 3.02, the integration has been good enough to produce the desired outputs.

7.13 Packaging

The executable user interface file and the OCR needed to be packaged. For this purpose, we created a folder and place three files in the folder. These files are the OCR scanner, traineddata and the user interface's executable file. After launching the user interface file, one can select the inputs and the process them via the OCR system. The outputs will be automatically generated in the package's folder. The user interface allows the users to name the output files.

Figure18 shows the package of the system.

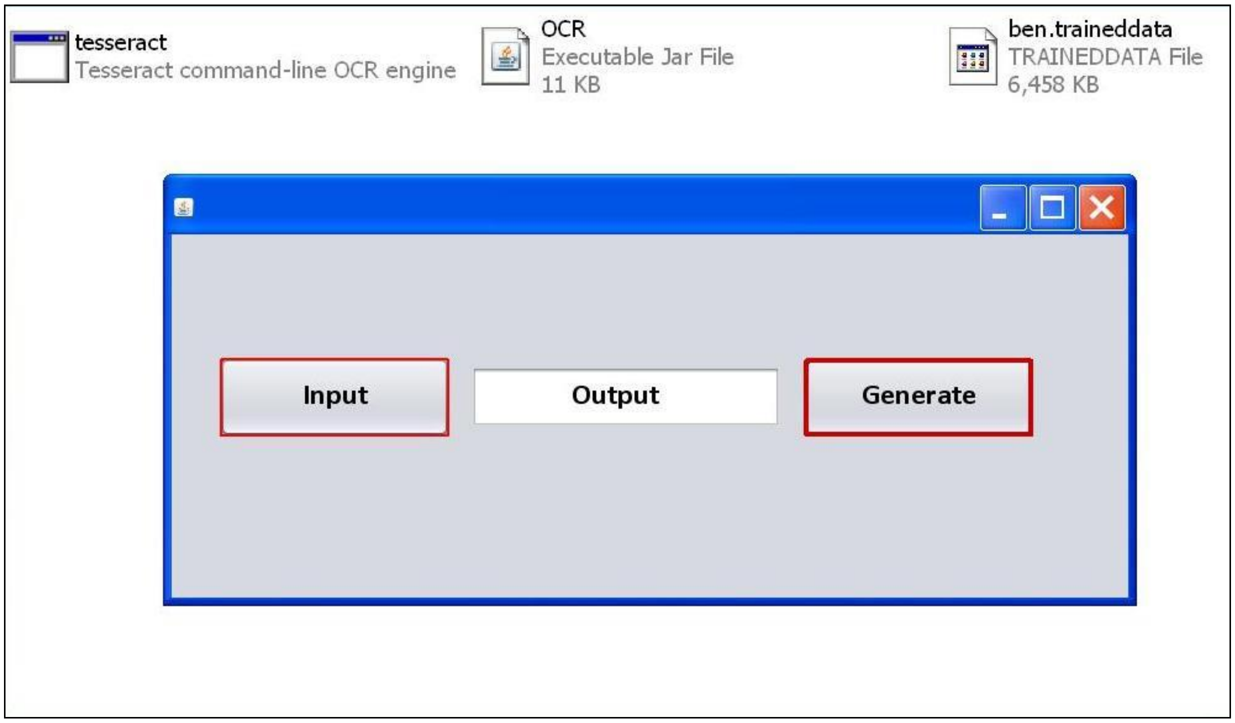


Figure18: System package

Chapter 8
ALGORITHM OF TESSERACT

Tesseract has a very sophisticated algorithm. It uses search and fetch policy for generating output. The images that we used for training are encoded into training files with 'tr' extension along with the box files. These tr files form a combined file named inttemp. This inttemp file along with other files form the concatenated traindata. So, when a character is read from the input file, Tesseract searches for a matching character in the images that are encoded into the inttemp file. If it finds a matching character, it gets the coordinates of the character in that image and uses that coordinates to find the required position in the box file. It fetches the character from the box file with that particular coordinate and writes it to the output file. If there is no matching character, it fetches the character that resembles most to the character of the input file. However, if the character set is too large, it may mismatch a character with another character and that's why accuracy of output does not always become full. Figure19 shows the block diagram of Tesseract's algorithm.

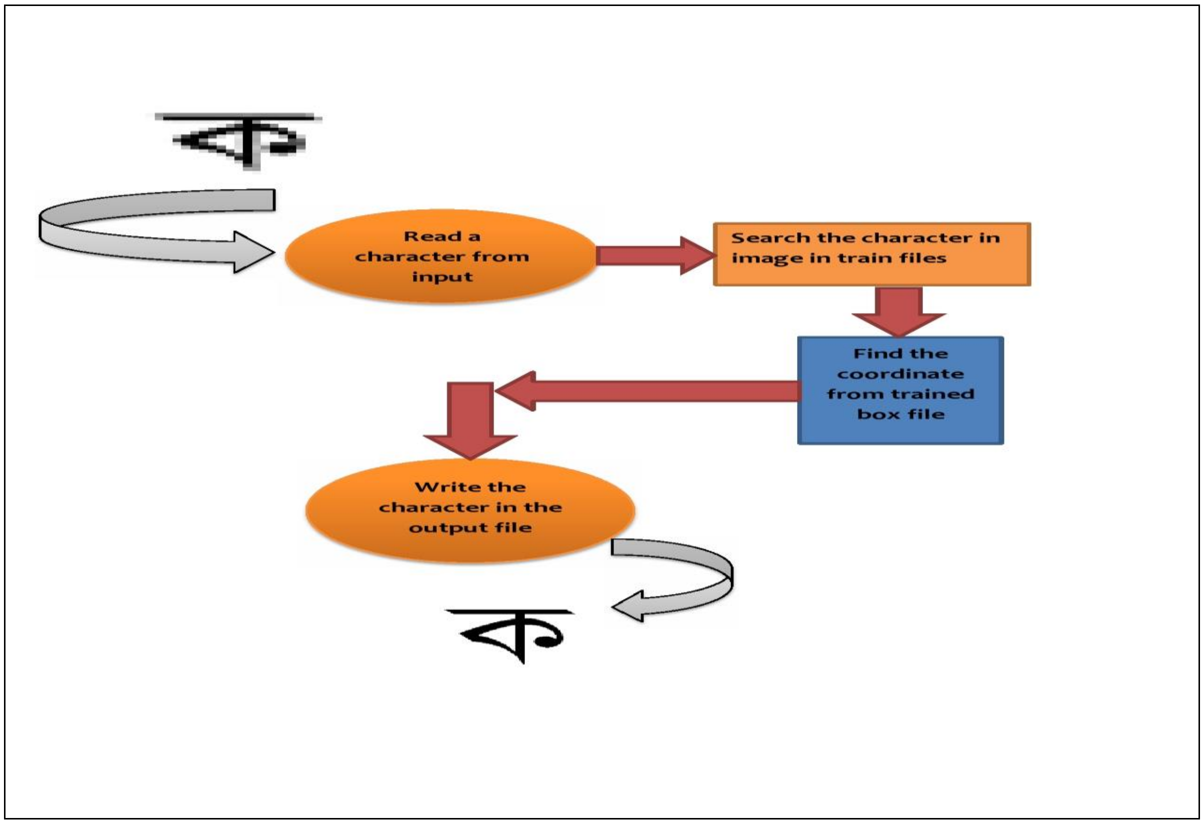

ক
Read a character from input
Search the character in image in train files
Find the coordinate from trained box file
Write the character in the output file
ক

Figure19: Block diagram of Tesseract's algorithm

CHAPTER 9
RESULT ANALYSIS

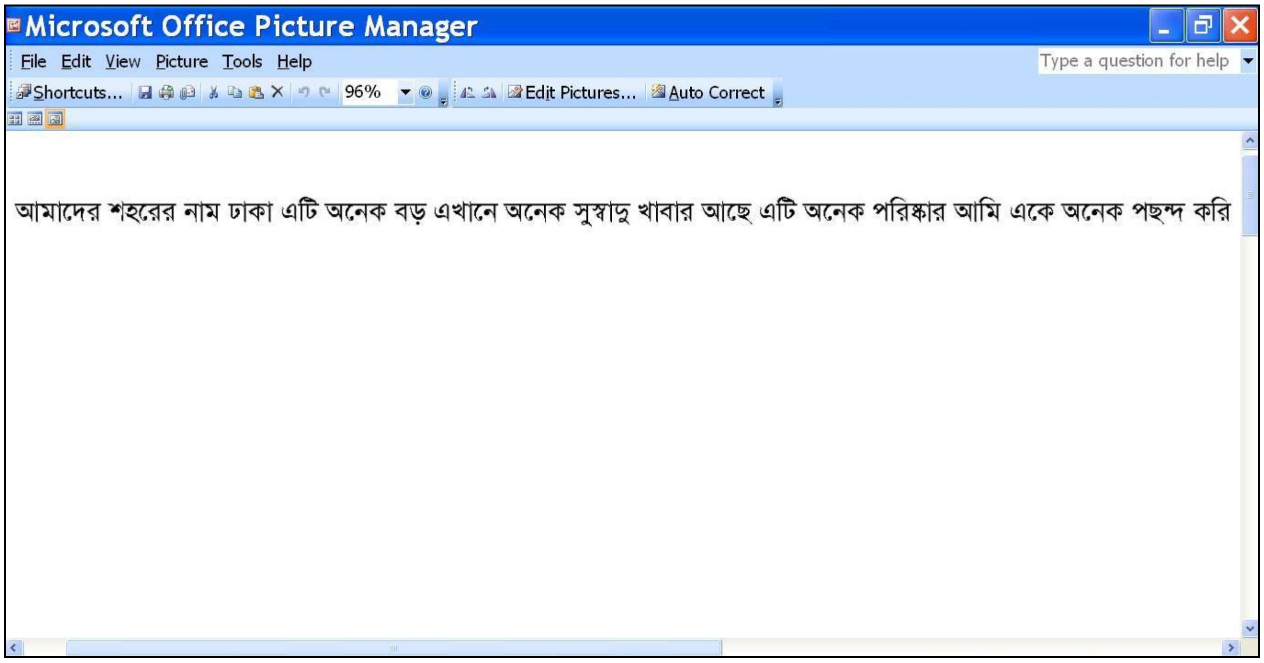
We tested the OCR with two different sets of inputs. One set of inputs contains converted images from text and the other set contains scanned images of printed documents. In each set, there were 100 inputs and around 25 words and 70 characters in each input. The OCR did not detect the spaces among words in various cases. We separated them manually to test the accuracy. We calculated the accuracy of the OCR with the following equations.

Accuracy rate (Character) = (Number of correct characters/Number of total characters) × 100%

Accuracy rate (Word) = (Number of correct Words/Number of total Words) × 100%.

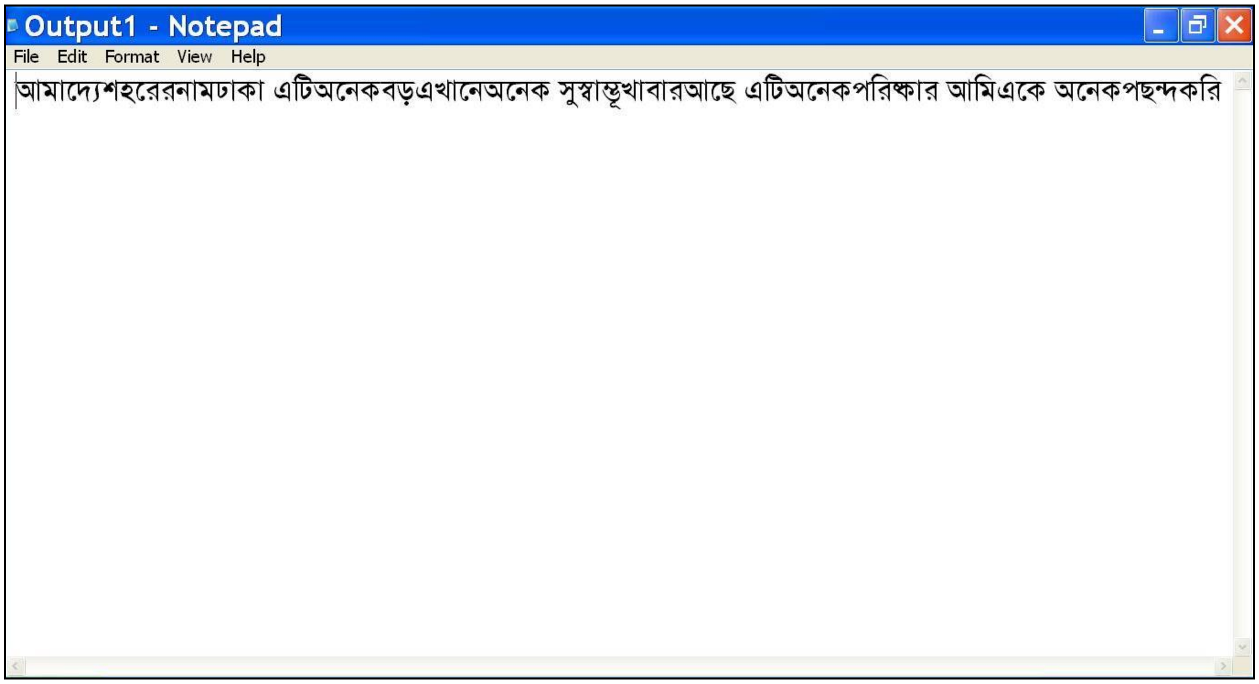
.We calculated accuracy rate based on characters and words separately. For scanned image file, the highest accuracy based on characters is 87.30% and the highest accuracy based on words is 40%. For image files that have been obtained by converting text files, the highest accuracy based on characters is 97.56% and the highest accuracy based on words is 90%.

Figure20 (a), Figure20 (b), Figure20(c), Figure20 (d), Figure20 (e), and Figure20 (f) show three samples of inputs and their respective outputs.



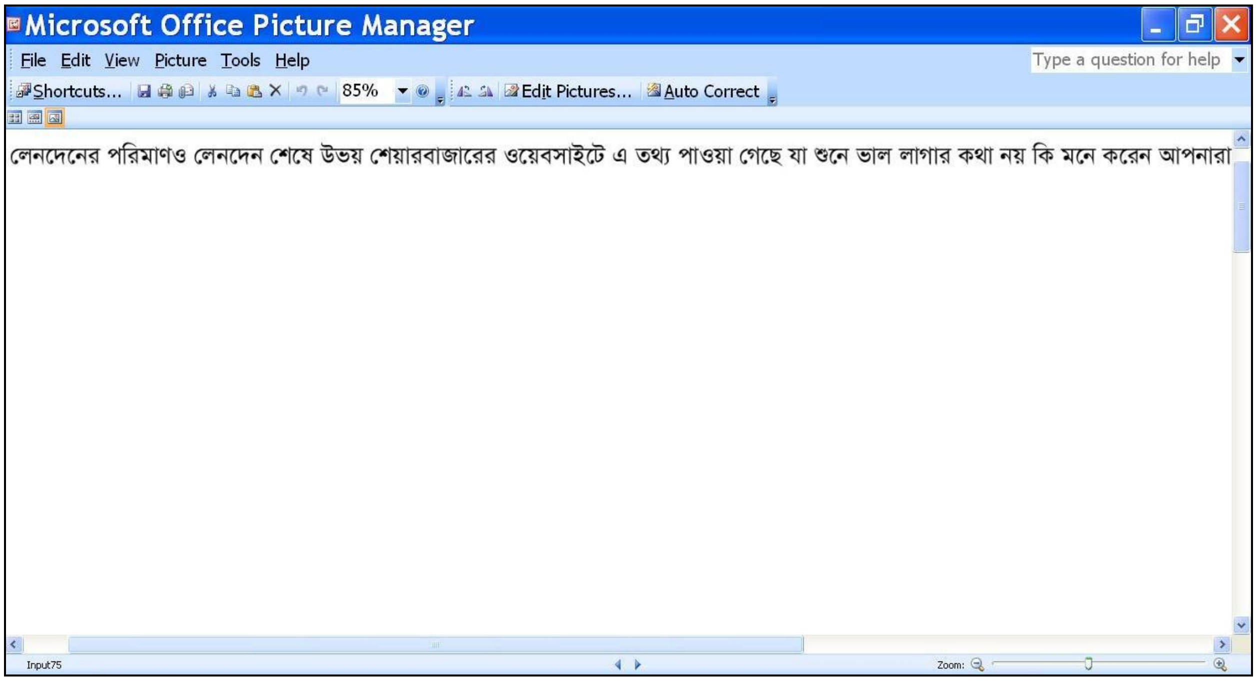
আমাদের শহরের নাম ঢাকা এটি অনেক বড় এখানে অনেক সুস্বাদু খাবার আছে এটি অনেক পরিষ্কার আমি একে অনেক পছন্দ করি

Figure20 (a): Sample Input1



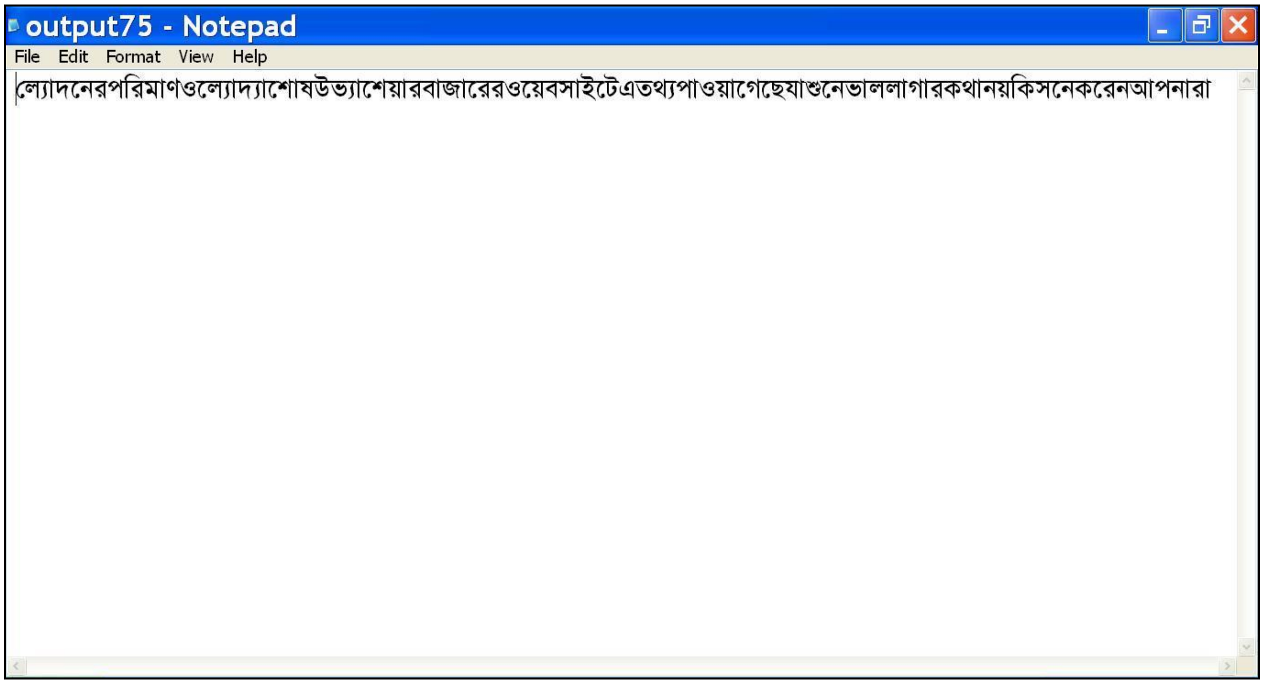
আমাদ্যেশহরেরনামঢাকা এটিঅনেকবড়এখানেঅনেক সুস্বাস্থূখাবারআছে এটিঅনেকপরিষ্কার আমিএকে অনেকপছন্দকরি

Figure20 (b): Output of sample Input1



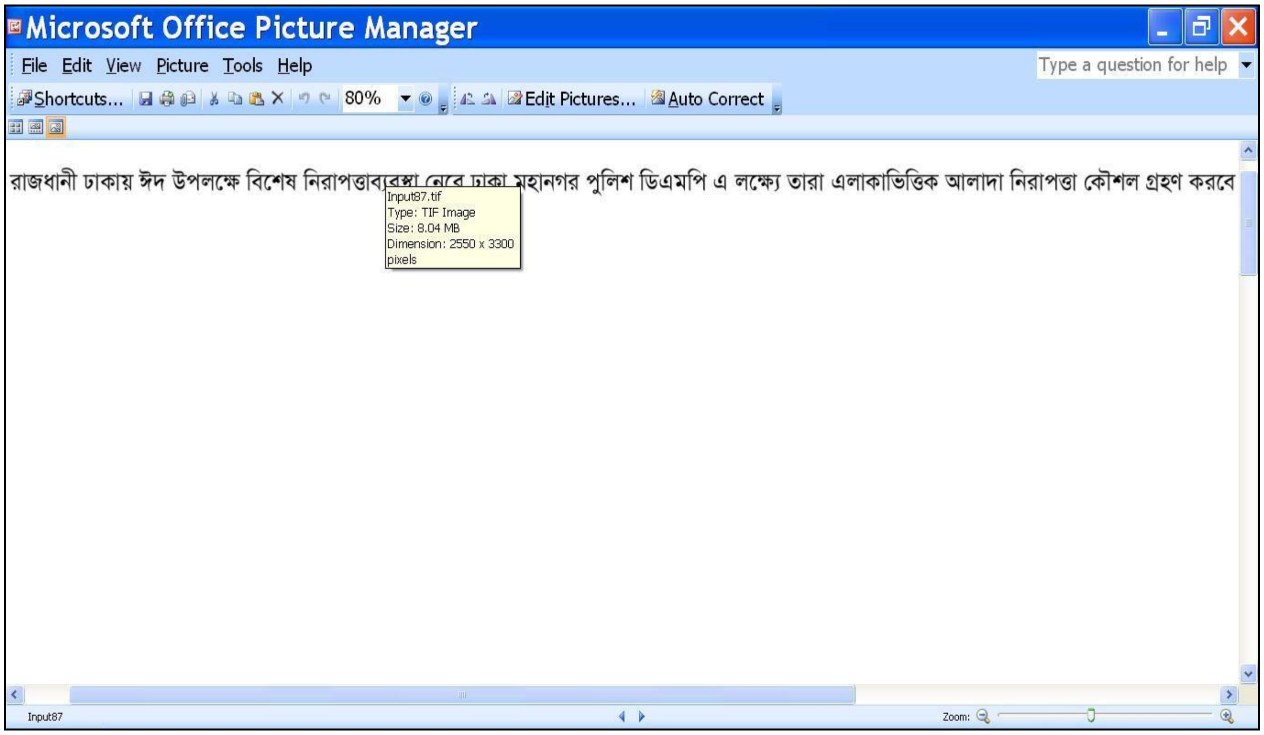
লেনদেনের পরিমাণও লেনদেন শেষে উভয় শেয়ারবাজারের ওয়েবসাইটে এ তথ্য পাওয়া গেছে যা শুনে ভাল লাগার কথা নয় কি মনে করেন আপনারা

Figure20 (c): Sample Input2



ল্যোদনেরপরিমাণওল্যোদ্যাশোষউভ্যাশেয়ারবাজারেরওয়েবসাইটেএতথ্যপাওয়াগেছেযাশুনেভাললাগারকথানয়কিসনেকরেনআপনারা

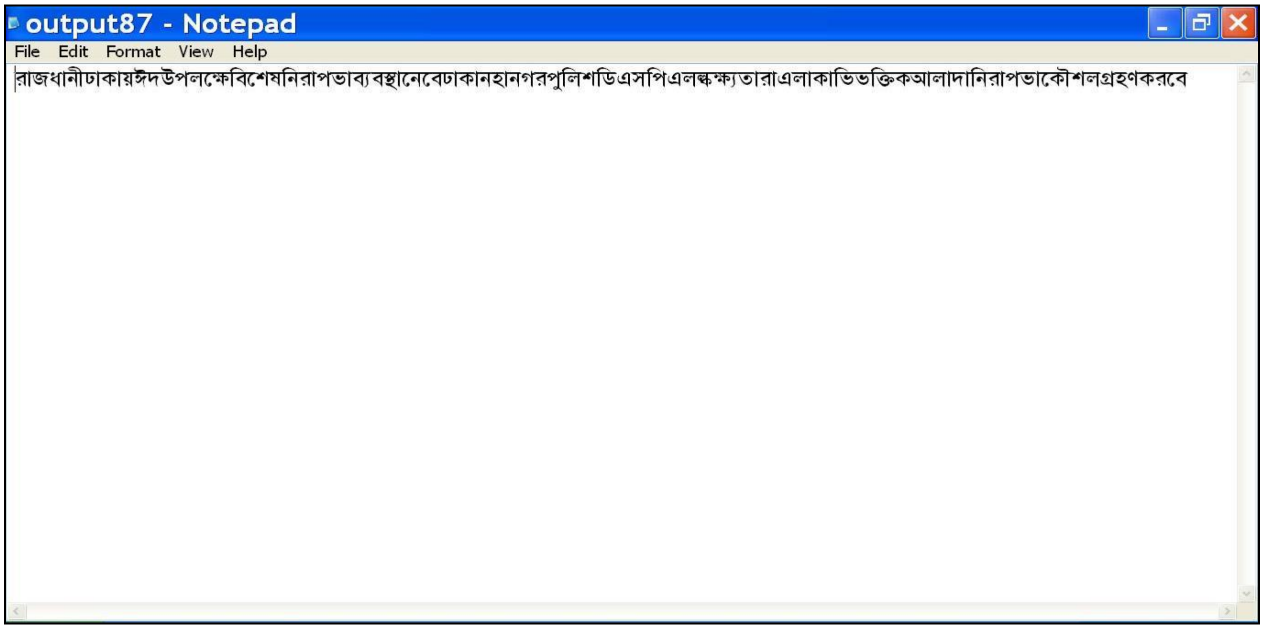
Figure20 (d): Output of sample Input2



রাজধানী ঢাকায় ঈদ উপলক্ষে বিশেষ নিরাপত্তাব্যবস্থা নেবে ঢাকা মহানগর পুলিশ ডিএমপি এ লক্ষ্যে তারা এলাকাভিত্তিক আলাদা নিরাপত্তা কৌশল গ্রহণ করবে

Input87.tif
Type: TIF Image
Size: 8.04 MB
Dimension: 2550 x 3300 pixels

Figure20 (e): Sample Input3



রাজধানীঢাকায়ঈদউপলক্ষেবিশেষনিরাপভাব্যবস্থানেবেঢাকানহানগরপুলিশডিএসপিএলন্ধক্ষ্যতারাএলাকাভিত্তিকআলাদানিরাপভাকৌশলগ্রহণকরবে

Figure20 (f): Output of sample Input3

Figure21 (a), Figure21 (b), Figure21 (c) and Figure21 (d) show the accuracies based on characters and words for converted and scanned images respectively.

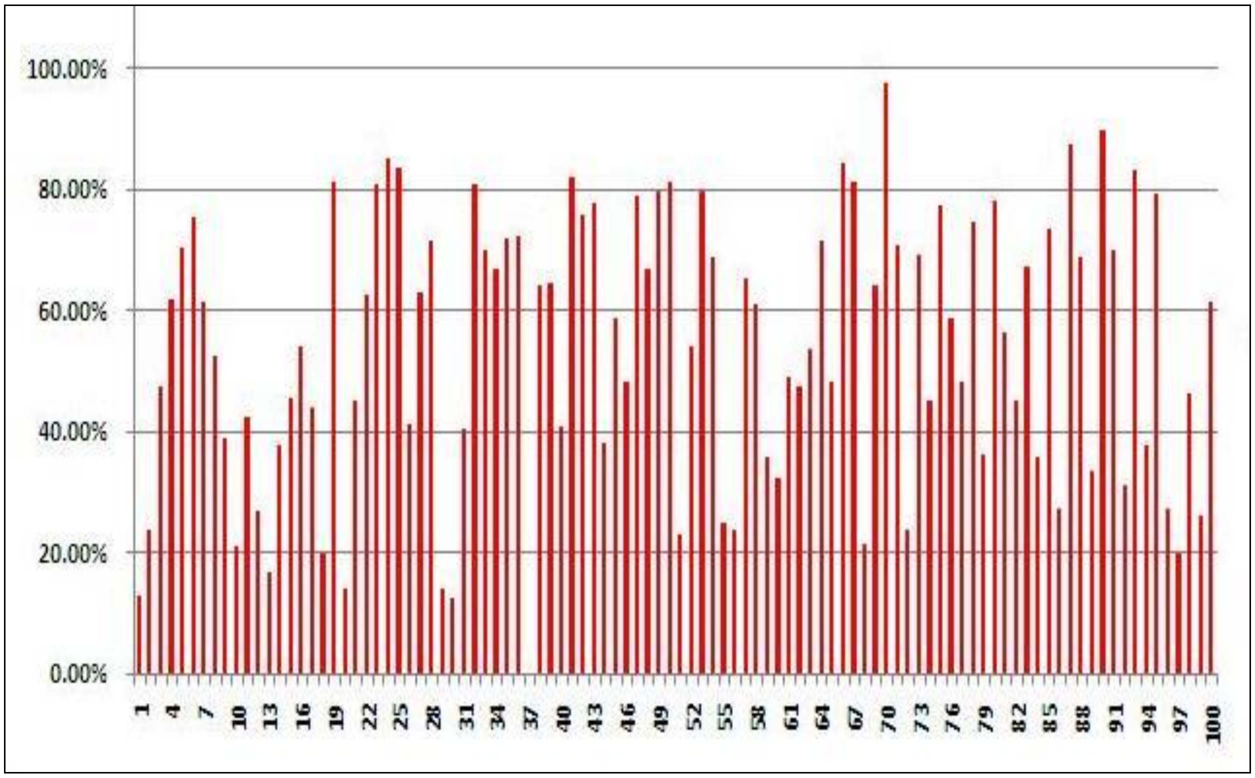


Figure21 (a): Accuracy of converted images based on character

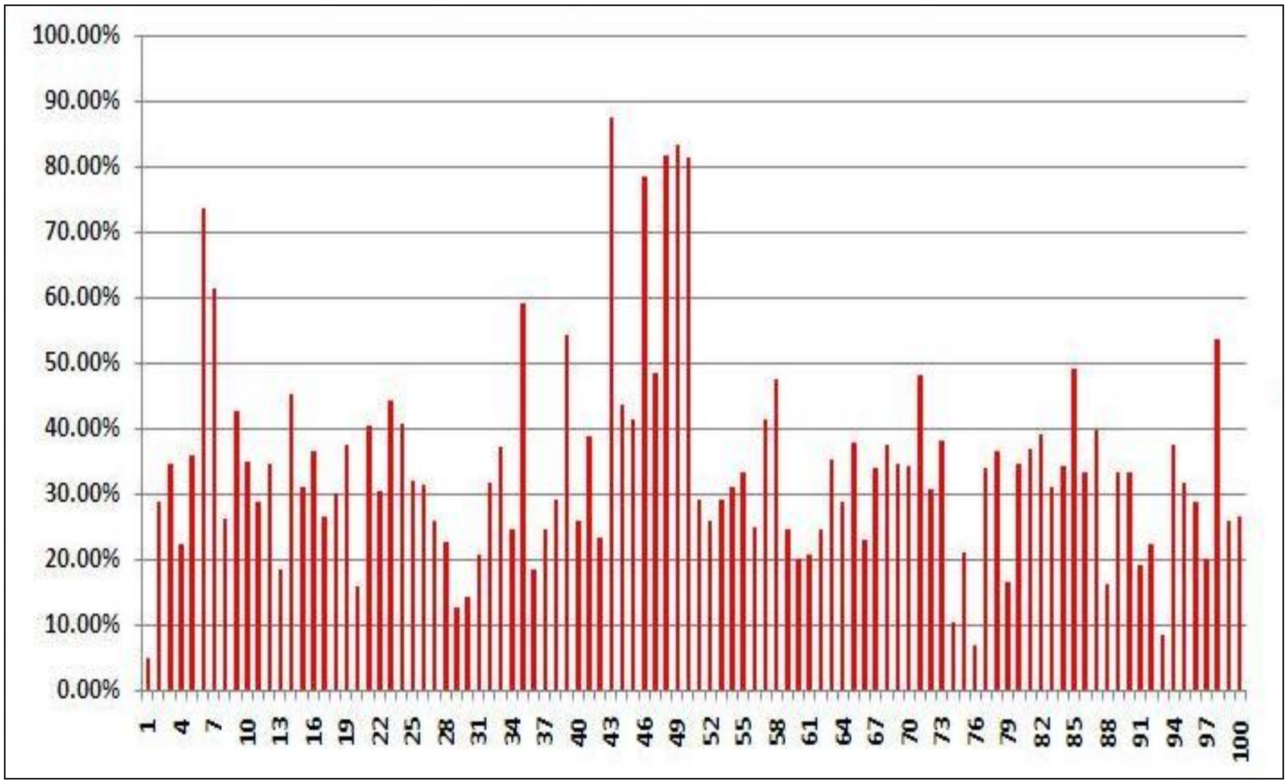


Figure21 (b): Accuracy of scanned images based on character

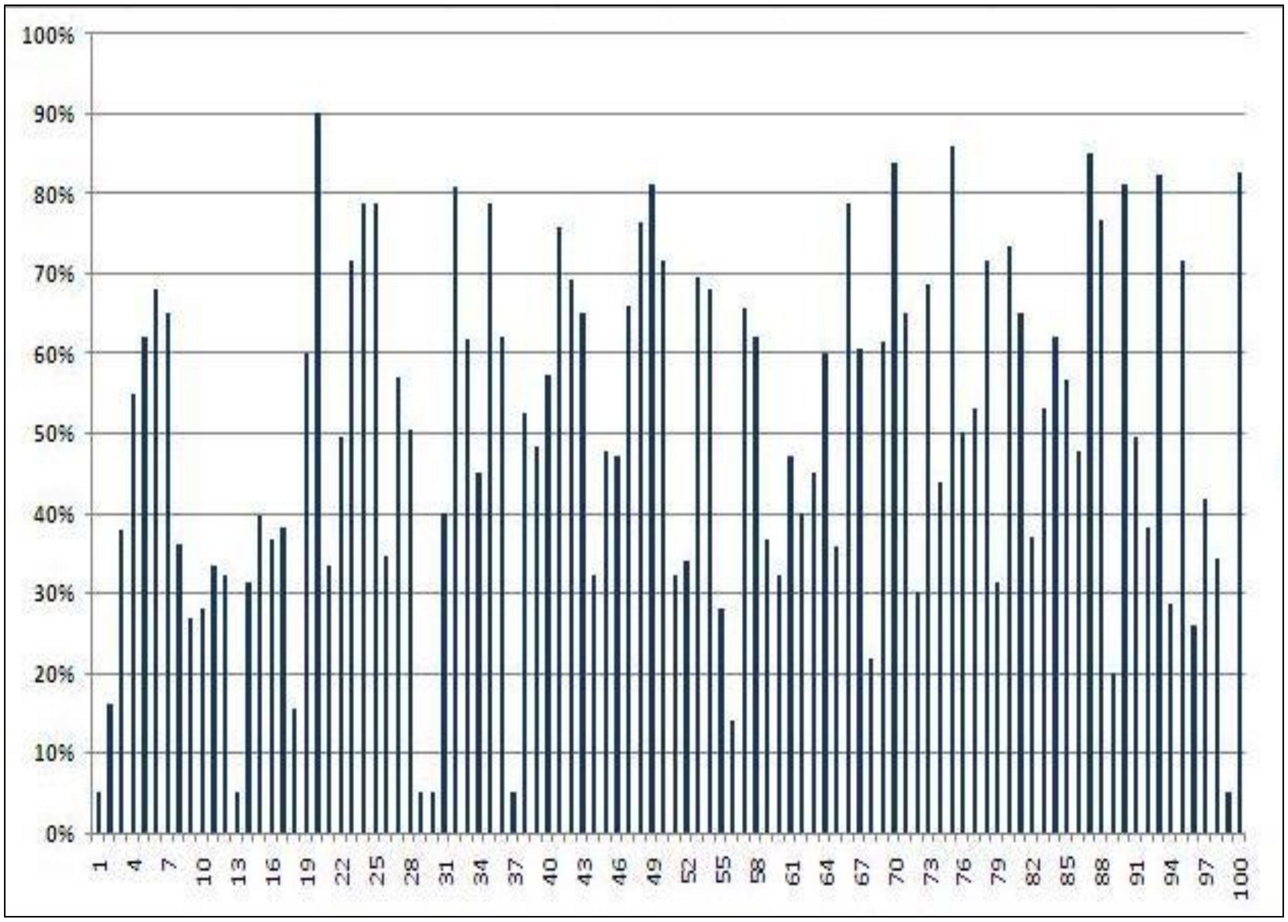


Figure21(c): Accuracy of converted images based on word

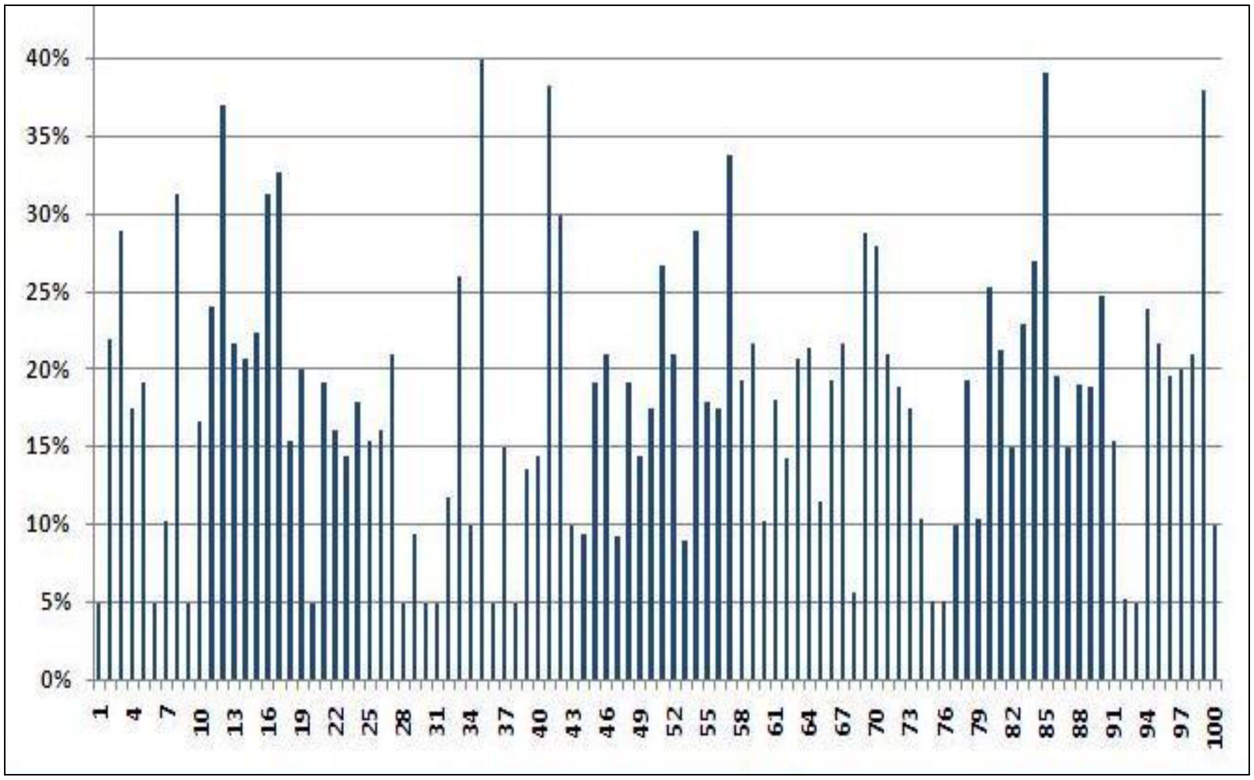


Figure21 (d): Accuracy of scanned images based on word

CHAPTER 10
CONCLUSION

We used ‘Solaimanlipi’ font to develop the character set for the OCR. So, the OCR would work fine on inputs that contain text in that font. However, if there are texts of other fonts available in the input, the OCR may mismatch to some extent. Though it is not quite possible to incorporate all fonts of Bengali language to the OCR due to execution speed limitation, we would integrate at least five more fonts to our OCR. These five fonts would be selected based on the diversity they possess so that all sorts of shapes for a particular character are covered by the OCR. The OCR takes ‘tiff’ and ‘png’ formatted images as inputs. So, if the inputs are not formatted in those two formats, it is necessary to convert them. The user can do it with third party software such as Paint. In future, we look forward to integrating a converter with our OCR. So, the user would be able to simply upload the input file, the OCR will automatically convert it if it is necessary and generate output. The OCR has been implemented using electronically converted images. Due to intricacy of the process, we could not use real images taken by camera or scanner from a printed file. We would use a few real images later to improve the OCR’s performance. We also plan to incorporate a Bengali to English translator with the OCR so that non-Bengali speakers can be benefitted. We also have a plan to add a spell-checker to improve word accuracy.

The OCR is implemented on research purpose. It has got its limits but it is an example of training an engine with expertise. We went through a lot of trial and error processes to find out which will be the best method to prepare images files from text and we finally used the dedicated java editor. Lots of efforts have also been put in modifying the box files. Java GUI is used for the simplicity of the software. We developed it as a desktop application so that it can be used offline. This project will be open for general purpose use for common users after adding further improvements.

REFERENCES

1. Abduallah, ArifBillah Al-Mahmud; Khan, Dr. Mumit Khan: A survey on script segmentation for Bangla OCR
2. Md. AbulHasnat, S. M. Murtoza, Dr. Mumit Khan: A high performance domain specific OCR for Bangla script
3. Hasnat, Md. Abul, Chowdhury, Muttakinur Rahman, Dr. Mumit Khan:Integrating Bangla script recognition support in tesseract OCR
4. S.Mahbub-Uz-Zaman, Tanjina Islam: Application of augmented reality: Mobile camera based bangla text detection and translation
5. Muttakinur Rahman Chowdhury (Shouro): Integration of Bangla script recognition support in OCRopus
6. http://en.wikipedia.org/wiki/Optical_character_recognition ***Last accessed on 15/08/2015***
7. https://en.wikipedia.org/wiki/Bengali_language ***Last accessed on 15/08/2015***
8. http://crblp.bracu.ac.bd/ocr.php ***Last accessed on 15/08/2015***
9. https://code.google.com/p/tesseract-ocr/wiki/Training ***Last accessed on 15/08/2015***
10. http://vietocr.sourceforge.net/training.html ***Last accessed on 17/08/2015***
11. https://www.omicronlab.com/bangla-fonts.html ***Last accessed on 16/08/2015***
12. www.sourceforge.net/p/vietocr/discussion/ ***Last accessed on 14/08/2015***
13. http://en.wikipedia.org/wiki/Language_code ***Last accessed on 14/08/2015***
14. http://www.succeed-project.eu/wiki/index.php/Tesseract_3.02 ***Last accessed on 18/08/2015***
15. http://en.wikipedia.org/wiki/ISO_639-3 ***Last accessed on 10/08/2015***
16. https://netbeans.org/kb/docs/java/gui-functionality.html ***Last accessed on 12/08/2015***
17. https://en.wikipedia.org/wiki/Image_file_formats ***Last accessed on 14/08/2015***
18. http://whatis.techtarget.com/definition/skewness ***Last accessed on 15/08/2015***
19. https://en.wikipedia.org/wiki/Image_noise ***Last accessed on 16/08/2015***
20. https://en.wikipedia.org/wiki/Tesseract_%28software%29 ***Last accessed on 20/08/2015***
21. https://en.wikipedia.org/wiki/NetBeans ***Last accessed on 21/08/2015***

APPENDIX

```
importjava.io.BufferedReader;

importjava.io.File;

importjava.io.IOException;

importjava.io.InputStreamReader;

importjavax.swing.JFileChooser;

public class Tesseract extends javax.swing.JFrame {

   String filename;

   String output;

private Object JTextField_path;

   /**

    * Creates new form Tesseract

    */

publicTesseract() {

initComponents();

   }

@SuppressWarnings("unchecked")

   // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-
BEGIN:initComponents

private void initComponents() {


        jButton1 = new javax.swing.JButton();

        jButton2 = new javax.swing.JButton();

        jTextField1 = new javax.swing.JTextField();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
jButton1.setFont(new java.awt.Font("Calibri", 1, 24)); // NOI18N
jButton1.setText("Input");
jButton1.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(255, 0, 0), 2, true));
jButton1.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEventevt) {
jButton1ActionPerformed(evt);
            }
        });

jButton2.setFont(new java.awt.Font("Calibri", 1, 24)); // NOI18N
jButton2.setText("Generate");
jButton2.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(204, 0, 0), 4, true));
jButton2.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEventevt) {
jButton2ActionPerformed(evt);
            }
        });

jTextField1.setFont(new java.awt.Font("Calibri", 1, 24)); // NOI18N
jTextField1.setText("               Output");
jTextField1.addActionListener(new java.awt.event.ActionListener() {
public void actionPerformed(java.awt.event.ActionEventevt) {
jTextField1ActionPerformed(evt);
            }
        });
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(39, 39, 39)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 186,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 250,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 19,
Short.MAX_VALUE)
            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 186,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(77, 77, 77))
    );
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(102, 102, 102)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 65,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 50,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
            .addContainerGap(133, Short.MAX_VALUE))
        );

pack();
    }// </editor-fold>//GEN-END:initComponents

private void jTextField1ActionPerformed(java.awt.event.ActionEventevt) {//GEN-
FIRST:event_jTextField1ActionPerformed
        // TODO add your handling code here:
    }//GEN-LAST:event_jTextField1ActionPerformed

private void jButton1ActionPerformed(java.awt.event.ActionEventevt) {//GEN-
FIRST:event_jButton1ActionPerformed
JFileChooser chooser=new JFileChooser();
chooser.showOpenDialog(null);
       File f=chooser.getSelectedFile();
filename=f.getAbsolutePath();
      // JTextField_path.setText(filename);// TODO add your handling code here:
    }//GEN-LAST:event_jButton1ActionPerformed

private void jButton2ActionPerformed(java.awt.event.ActionEventevt) {//GEN-
FIRST:event_jButton2ActionPerformed
output =jTextField1.getText();
callExec(filename);  // TODO add your handling code here:
    }//GEN-LAST:event_jButton2ActionPerformed

    /**
    * @paramargs the command line arguments

    */

public static void main(String args[]) {

      /* Set the Nimbus look and feel */

      //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

      /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

       * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

       */

try {

for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

            }

          }

      } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Tesseract.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

      } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Tesseract.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

      } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Tesseract.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

      } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Tesseract.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        }
        //</editor-fold>

        /* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
public void run() {
newTesseract().setVisible(true);
            }
        });
    }
voidcallExec(String fileName){
        String s = null;

try {

        // run the Unix "ps -ef" command
            // using the Runtime exec method:
            Process p = Runtime.getRuntime().exec("tesseract "+fileName+" "+output+" -l ben");

BufferedReaderstdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));

BufferedReaderstdError = new BufferedReader(new
InputStreamReader(p.getErrorStream()));

            // read the output from the command
```

```
System.out.println("Here is the standard output of the command:\n");
while ((s = stdInput.readLine()) != null) {
System.out.println(s);
            }


            // read any errors from the attempted command
System.out.println("Here is the standard error of the command (if any):\n");
while ((s = stdError.readLine()) != null) {
System.out.println(s);
            }


System.exit(0);
        }
catch (IOException e) {
System.out.println("exception happened - here's what I know: ");
e.printStackTrace();
System.exit(-1);
        }    // TODO code application
    }


    // Variables declaration - do not modify//GEN-BEGIN:variables
privatejavax.swing.JButton jButton1;
privatejavax.swing.JButton jButton2;
privatejavax.swing.JTextField jTextField1;
    // End of variables declaration//GEN-END:variable
```