

Urdu OCR

Optical Character Recognition for Urdu Language



By

Ayesha Tahir Khan Khattak
Manahil Ahmed
Ismaeel Moeen

Project Supervisor

Assistant Professor Bilal Rauf

Submitted to the Faculty of Computer Software Engineering Department,
Military College of Signals, National University of Sciences and Technology, Islamabad
In partial fulfillment for the requirements of a B.E. Degree in Software Engineering

July 2020

DECLARATION OF ORIGINALITY

We hereby declare that no segment of this work in this proposition has been submitted for the benefit of another award or qualification in either this institution or anyplace else. This thesis report has not been formerly published in any structure nor does it include any verbatim of the published resources which could be treated as violation of the international copyright decree. We also affirm that we do recognize the terms 'plagiarism' and 'copyright' and that in case of any copyright infringement or plagiarism established in this thesis, we will be held fully accountable of the consequences of any such violation.

ABSTRACT

In the running world, there is developing need and urge for the systems which help to perceive characters or traits in computer when useful data is screened through physical printed paper books/records as we understand that we have a number of daily papers, books and articles which are in hard back or printed structure. In these modern and advancement times there is a huge demand in putting away the details or data useful in these paper archives into a digitized form i.e. in computer memory and when needed, utilizing the data. OCR for the Urdu language is being made for digitizing the hard-back physical Urdu text and later have the ability to edit the digital data.

OCR is a worldwide strategy planned for recognizing text in pictures. It converts or changes over manually written, self-made or printed text into data that can be altered by a client on a PC. We are proposing making an OCR software for Urdu that would convert Urdu text in pictures into an editable content document.

Machine-readable text would then be decoded by screen readers, tools and apps that use text to audio feature so to read out the words on a screen so visually impeded individuals can be able to understand what's written on the print out.

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis titled “Urdu OCR” (Optical Character Recognition for Urdu Language), carried out by Ayesha Tahir, Manahil Ahmed and Ismaeel Moeen under the supervision of Assistant Professor Bilal Rauf for partial fulfillment of Degree of Bachelors of Software Engineering, in Military College of Signals, National University of Sciences and Technology during the academic year 2019-2020 is correct and approved. The plagiarism is [12 %](#).

Approved By

Supervisor
Assistant Professor Bilal Rauf
CS Dept, MCS

Dated: July 2020

In The Name of Allah, the Most Benevolent, the Most Merciful.

DEDICATION

This research is lovingly dedicated to our parents, well-wishers and most of all our Supervisor Sir Bilal Rauf, without whose unstinting cooperation and unflinching support, made it possible for such a work of this magnitude to be done.

ACKNOWLEDGEMENTS

All praises for Allah Almighty who gave us the strength to accomplish this mighty and helped us to succeed this task despite numerous difficulties and hardships on our way.

We offer our utmost gratitude to our Supervisor Sir Bilal Rauf for his steady direction and encouragement. The OCR would not have been achievable without his expertise, unfailing persistence and invaluable efforts.

We revere the moral support and passionate encouragement of Sir Ahmed Muqem Sherri (Dept of CS, MCS-NUST) throughout the development of our project.

TABLE OF CONTENTS

ABSTRACT.....	3
1.0 Introduction.....	14
1.1 Project Overview:	15
1.2 Problem Statement:	15
1.3 Objectives	15
1.4 Approach/ Research Methodology.....	16
1.4.1 Uploading Image:	16
1.4.2 The OCR Process:	16
1.4.2.1 Distinguishing between text and images - Segmentation.....	16
1.4.2.2 Character recognition – Feature Extraction	17
1.4.2.3 Recognition of Words	17
1.4.3 Creating Text File.....	17
1.4.4 Saving Text File	17
1.5 Limitations.....	17
1.6 Organization of the Document.....	18
2.0 Literature Review.....	19
2.1 Project Domain:	20
2.2 Literary Review:	20
2.2.1 Features of the Urdu Language:	21
2.2.2 Problems of Urdu Script:	24
2.3 Other Problems in Urdu OCR:	25
2.4 Characteristics of Urdu Characters.....	26
2.5 Existing Solution and their flaws:	28

2.6	Issues addressed by Urdu OCR:	29
2.7	Novelty of Urdu OCR:	29
2.8	Summary:	30
3.0	Technological Requirements.....	31
3.1	Hardware Requirements.....	32
3.1.1	Laptop Computer:	32
3.2	Software Requirements.....	33
3.2.1	Ubuntu.....	33
3.2.2	PyCharm.....	33
3.2.3	Python Programming Language.....	33
3.2.4	Google Tesseract	34
3.2.5	Open CV	34
3.3	Operating System Requirements.....	35
3.4	Setting up Working Environment.....	38
3.4.1	Installing the Necessary Libraries.....	42
4.0	Methodology.....	43
4.1	Tesseract Training.....	44
4.2	Image Pre-processing	48
4.3	Recognition.....	49
4.4	File Creation.....	50
5.0	Results and Analysis.....	52
5.1	Final Deliverable.....	53
5.2	Image Editing Results.....	54
5.3	Recognition Results	56
5.4	File Creation	60
6.0	Conclusion & Possible Future Work.....	62
6.1	Future Work.....	63

2

6.2	Conclusion	64
	References	65
	Appendix A	69
	Appendix B	71
	Appendix C	72

LIST OF FIGURES

Figure 2-1 Working set of Urdu Character	21
Figure 2-2 Test Image	28
Figure 3-1 Laptop Specifications	32
Figure 3-2 Ubuntu LTS with swap file	35
Figure 3-3 RAM.....	36
Figure 3-4 Disk Properties	37
Figure 3-5 Ubuntu Linux Installation Menu	38
Figure 3-6 Custom partitioning option selection	39
Figure 3-7 Partition creation task completed	39
Figure 3-8 Credentials put up	40
Figure 3-9 Log in page.....	40
Figure 3-10 Successful Tesseract installation.....	41
Figure 4-1 Python tesstrain.sh script.....	45
Figure 4-2 Python code for extracting LSTM.....	45
Figure 4-3 Python script for evaluating LSTM.....	46
Figure 4-4 Python script for Fine tuning LSTM	46
Figure 4-5 Evaluation Results	47

Figure 4-6 Python command for Uploading Image	48
Figure 4-7 Geometry for Crop Feature	48
Figure 4-8 Python command for Crop Feature	48
Figure 4-9 Python command for Rotation	49
Figure 4-10 Extract Text Function.....	49
Figure 4-11 OCR Command for Image Conversion.....	50
Figure 4-12 OCR Command for Saving File.....	50
Figure 4-13 Directory for Saving.....	51
Figure 5-1 Final Deliverable.....	53
Figure 5-2 Select Image Dialogue Box.....	54
Figure 5-3 Edit Image Screen	54
Figure 5-4 Rotating Image with Degrees.....	55
Figure 5-5 Cropping the input image.....	55
Figure 5-6 Cropping desired portion of image.....	56
Figure 5-7 Multi-line Naskh test image	57
Figure 5-8 Recognition results.....	57
Figure 5-9 Recognition results (continued)	58
Figure 5-10 Anjali Old Lipi test image.....	59

Figure 5-11 Recognition Results	59
Figure 5-12 Recognition results (continued)	60
Figure 5-13 Creating .txt file dialogue box	61
Figure 5-14 Text file saved in the user selected directory.....	61

List of Tables

Table 2-1 Graphical representation of Urdu alphabet.....	22
Table 2-2 Graphical representation of similar Urdu characters	24
Table 2-3 Existing solutions and their flaws	28
Table 2-4 Comparison of Urdu OCR with existing solutions.....	30

Chapter 1: Introduction

1.1 Project Overview

1.2 Problem Statement

1.3 Objectives

1.4 Approach

1.5 Limitations

Introduction

This chapter provides comprehensive introduction of the project titled “Urdu OCR”

1.1 Project Overview:

Optical Character Recognition (OCR) is a software used for the conversion of text in images that can be written, typed or printed and converted into text that can be edited by computer. OCR could be an area of research in artificial intelligence, pattern recognition and machine vision. An OCR system empowers us to require a book or an article in a magazine, explicitly feed it into an electronic database record, and then edit the record using a word processor.

The project aims to create an OCR for Urdu-based scripts. If we wanted a magazine article or a written document to be digitized, we could spend hours retyping and then fixing misprints. Or, in few minutes, we could convert all the necessary materials to digital format using Optical Character Recognition.

1.2 Problem Statement:

Urdu OCR is required to convert the printed Urdu Books to editable text files. Recent work in this field has been able to advance some modern techniques to address the difficulty of Urdu styles of writing. Nevertheless these methods have not been checked for all Urdu letter characters. Therefore, we need a program that can handle all Urdu text classes and recognize characters among those classes. We will need a full review of the segmentation techniques to build a segmentation-based Urdu OCR.

1.3 Objectives:

In this part of the world, Urdu Optical Character Recognition provides us with a new possible way to realize the vision of the mode of communication among man and machine. It is going to develop and extend existing knowledge to new and modern skylines. Centuries old script that is uncommon in Urdu will become easily available to common man. The main aim of recognizing character is to assist the visually impaired and simulate human reading capabilities.

Following are the main objectives of our project:

1. Digitizing paper documents.
2. Automated Data Entry.
3. Assisting the visually impaired.
4. Saving time of manually entering printed data onto a computer.
5. Saving space as reams of paper documents will be converted to digital text files.

1.4 Approach/ Research Methodology

This section summarizes the approach used by Urdu OCR to convert Urdu text image into editable text file.

1.4.1 Uploading Image

This step involves uploading an Urdu text image. There are multiple image formats used for this purpose i.e. JPEG, BMP, PNG and JPG.

The image can be edited which includes cropping and rotating the image by degrees.

1.4.2 The Distinguished OCR Process

The main voluntary steps involved in OCR are as following.

1.4.2.1 Distinguishing between text and images – Segmentation

In this step, the text of the selected image is identified. The text image shall be read and then possible characters, words and lines shall be segmented.

1.4.2.2 Character recognition – Feature Extraction

Characters are recognized in this step using a process known as feature extraction.

This step involves applying various techniques that are used for text extraction to the segmented image. This is done to extract/obtain the major details and to recognize the Urdu script in the images.

1.4.2.3 Recognition of Words or Script

Following the procedure of character recognition, word recognizable proof is finished by contrasting the series of characters with our prepared dataset.

1.4.3 Creating Text File

Once the text is recognized, the software shall create an editable file with .txt format.

1.4.4 Saving Text File

The final step involves saving the output onto the device through clicking on the save button.

1.5 Limitations:

1. OCR may not convert characters with exceptionally huge or little text sizes.
2. The printed text without skew is the main focus of the application because of time constraint.
3. There are important non-textual characters or glyphs that do not get converted to characters by OCR.
4. OCR will work only with one particular Urdu font.

1.6 Organization of Document

- Chapter 1** Provides essential details and overview of Urdu OCR
- Chapter 2** Deals with the literature review carried out for Urdu OCR and discusses its core features.
- Chapter 3** Discusses the hardware and software requirements of Urdu OCR, including technical specifications and setup for working environment
- Chapter 4** Discusses the methodology followed by Urdu OCR in detail.
- Chapter 5** Provides all results, testing and analysis regarding the main working of Urdu OCR
- Chapter 6** Presents overview of enhancements and future work

Chapter 2: Literature Review

2.1 *Project Domain*

2.2 *Literature Review*

2.3 *Other problems in Urdu OCR*

2.4 *Characteristics of Urdu Characters*

2.5 *Existing Solutions and their Flaws*

2.6 *Issues addressed by Urdu OCR*

2.7 *Novelty of Urdu OCR*

2.8 *Summary*

Literature Review

This chapter deals with comprehensive details of Urdu language, OCR tools and solutions offered worldwide, their limitations and the novelty of our solution.

2.1 Project Domain:

The software solution for changing typed, manually written or photocopied text into editable text that is readable by the machine. With OCR, a large number of paper-based documents, in all languages and myriad formats can be translated into machine-readable text that will not only make storage easier but will also make data previously digitally inaccessible to anyone, now easily accessible with a click. Imagine the number of archive boxes filled with city or state-owned paper. Such images and text can be scanned through the OCR Technology and converted to text documents so as to convert the hard-printed realm to the digital world. Also, the visually impaired people in our country have great difficulty in hiring an assistant to read Urdu newspapers / books. It costs them a bulk too. Uneducated people on the other hand do not understand a single piece of printed Urdu text. Urdu OCR is a low-cost solution to the problems described above. OCR Technology simplifies the task of manually inputting Urdu data, making space for stacking the backdrops of printed papers and, aiding blind individuals through Urdu translators working with text.

2.2 Literature Review:

As we know that the national language of Pakistan is Urdu, it's one of the significant contents spoken in the Indian subcontinent and it is advanced in the subcontinent from the amalgamation of Arabic, Turkish, Farsi and Hindi Languages. Urdu has a 58-character set characterized by National Language Authority Pakistan as it is being displayed in figure 2.1. In any case, significant detail here is that only 40 essential characters and one do-chashmi-hay is utilized to frame every single composite letters in order. Along these lines, the total working set comprises of just 41 characters.

ا ب پ بھ پ پھ ت تھ ٹ ٹھ ث ج جھ چ چھ ح خ د دھ ڈ ڈھ ذ ر رھ ز زھ
 ش س ص ض ط ظ ع غ ف ق ک گ گل لھ م مھ ن نھ ہ و وھ ہ ق ہ
 کی بے۔

Figure 2.1: Complete working set of Urdu Characters

The Urdu alphabet is composed from right to left in the Urdu language. It is an alteration of the Persian letters in order, which is itself a subsidiary of the Arabic letters in order. Urdu is normally written in the calligraphic Nasta'liq and Naskh content, while Arabic is all the more ordinarily in the Naskh style. Normally, uncovered transliterations of Urdu into Roman letters exclude numerous phonemic components that have no equal in English or different dialects usually written in the Roman alphabets. National Language Authority of Pakistan has built up various frameworks with explicit documentations to connote non-English sounds, however these can only be appropriately perused by somebody effectively acquainted with Urdu, Persian, or Arabic for letters, for example, غ ط or ص ق ط and Hindi letters, for example, ङ.

2.2.1 Features of the Urdu Language

As opposed to the English content, there is no capital or little characters in Urdu, yet the last character of a word can be considered as a capital character as a great part of the time, it presents the full form of the character and the characters at starting and middle positions are considered as small. Each character has an autonomous shape other than different joining structures, anyway a bit of the letters like the characters making the word Urdu (اردو) or of the similar class are not join able or can't be associated. Urdu letters in order utilizes consonant letters, diacritic imprints, numerals, accentuation, vowels, and two or three superscripts signs.

The graphical portrayal of each character has more than one structure dependent upon its area and setting in the word. For the most part, every character has four structures that is beginning, middle, last and independent as appeared in table-2.1.

Table 2.1: Graphical Representation of Urdu Haroof

#	Char	Forms	Name	
	حرف	اشکال	اسم	Name
0	ء		ہمزہ	hamzah
1	ا	ا	الف	alif
1a	آ	آ	الف مدّ	alif madd
2	ب	ببب	بے	bē
2h		بہ بہ	بھے	bhē
3	پ	پپپ	پے	pē
3h		پہ پہ	پھے	phē
4	ت	تتت	تے	tē
4h		تہ تہ	تھے	thē
5	ٹ	ٹٹٹ	ٹے	t.ē
5h		ٹہ ٹہ	ٹھے	t.hē
6	ث	ثثث	ثے	sē
7	ج	ججج	جیم	jīm
7h		جہ جہ	جھے	jhē
8	چ	چچچ	چے	čē = cē
8h		چہ چہ	چھے	čhē = chē
9	ح	ححح	بڑی ہے	bar.ī Hē
10	خ	خخخ	خے	xē = khē
11	د	د	دال	dāl
11h		دھ دھ	دھے	dhē
12	ڈ	ڈ	ڈال	d.āl
12h		ڈھ ڈھ	ڈھے	d.hē
13	ذ	ذ	ذال	zāl
14	ر	ر	رے	rē

#	Char	Forms	Name	
	حرف	اشکال	اسم	Name
14h		رہ	رہے	rē
15	ڑ	ڑ	ڑے	r.ē
15h		ڑھڑھ	ڑھے	r.hē
16	ز	ز	زے	zē
17	ڑ	ڑ	ڑے	žē = zhē
18	س	سس	سین	sīn
19	ش	ششش	شین	šīn = shīn
20	ص	صصص	صاد	Sād, Suād
21	ض	ضضض	ضاد	Zād, Zuād
22	ط	ططط	طوے	Tōē
23	ظ	ظظظ	ظوے	Zōē
24	ع	ععع	عین	'ain
25	غ	غغغ	غین	ġain
26	ف	ففف	فے	fē
27	ق	ققق	قاف	qāf
28	ک	ککک	کاف	kāf
28h		کھکھ	کھے	khē
29	گ	گگگ	گاف	gāf
29h		گھگھ	گھے	ghē
30	ل	للل	لام	lām
30h		لھ	لام	lām

#	Char	Forms	Name	
	حرف	اشکال	اسم	Name
31	م	مم	میم	mīm
31h		مہ	میم	mīm
32	ن	نن	نون	nūn
32h		نہنہ	نون	nūn
32a	ں	ں	نون غنہ	nūn-e gunnah
32ah		ں	نون غنہ	nūn-e gunnah
33	و	و	واو	vāo
33h		وہ	واو	vāo
34	ہ	بہ	چھوٹی ہے	chōt.I hē
34b	ہ	ہہہ	دوچشمی ہے	dō-čāsmī hē
35	ی	یی	چھوٹی ہے	chōt.I yē
35a	یہ	یہی	چھوٹی ہے	chōt.I yē
35b	ے	ے	بڑی ہے	bar.I yē

2.2.2 Problems of Urdu Script

In spite of a huge character set, Urdu/Arabic has a little arrangement of characters which are effectively distinguishable from each other. The rest of the character differ from these character using specks/dots/symbols above or underneath these shapes. The table 2.2 shows group of similar characters and their derived forms.

Table 2.2: Graphical Representation of similar Urdu characters

S.#	Characters			S.#	Characters		
	Standalone	Forms	Groups		Standalone	Forms	Groups
1	ء		1	21	ص	ص ص ص	7
2	ا	ا	2	22	ض	ض ض ض	
3	ب	ب ب ب	3	23	ط	ط ط ط	
4	پ	پ پ پ		24	ظ	ظ ظ ظ	8
5	ت	ت ت ت		25	ع	ع ع ع	9
6	ٹ	ٹ ٹ ٹ	26	غ	غ غ غ		
7	ث	ث ث ث	27	ف	ف ف ف		
8	ج	ج ج ج	4	28	ق	ق ق ق	10
9	چ	چ چ چ		29	ک	ک ک ک	11
10	ح	ح ح ح		30	گ	گ گ گ	
11	خ	خ خ خ	31	ل	ل ل ل	12	

12	د	د	5	32	م	مم	13
13	ڈ	ڈ		33	ن	ذن	14
14	ذ	ذ		34	ں	ں	
15	ر	ر		35	و	و	15
16	ڑ	ڑ		36	ہ	ہہ	16
17	ز	ز		37	ہ	ہہہ	17
18	ژ	ژ		38	ھ	ھہہہ	18
19	س	سس		6	39	ی	یی
20	ش	شش	6	40	ئ	ئی	20
				41	ے	ے	21

As shown in the tables above, only 21 groups exist out of the 41-character set. It will further confound the recognition phase of Urdu characters. Further the study of other structures (beginning, center and final) of these character uncovers that ein (ع) is like hamza (ء), wow (و) may be mistaken for (ی), ze (ز) resembles noon (ن) and (ذ), dhal (ڈ) is close match to tay (ٹ) and mem (م) can be mistaken for center form of ein (ع) and with the independent he (ہ). A key contrast between Latin script and Arabic script is the way that numerous letters only vary by a dot(s) however the essential stroke is actually the same.

2.3 Other Problems in Urdu OCR

The following is a rundown of issues found in Urdu characters yet it's by one way or another regular to all dialects which are utilizing Arabic content, for example Persian, Punjabi, Sindhi, Hindko, Pushto Siraiki, Balochi. As holy Book of Muslims is Quran and it's in Arabic Language so it's easy for the Muslims to read Arabic. Despite the fact that, Arabic content hasn't gotten enough consideration of the analysts. Little exploration progress has been accomplished subsequent to contrasting with the one done on the Latin and Chinese. The arrangements that are accessible in the market are still a long way from being great. The reasons which prompted this outcome are as per the following:

- Government provides no financial support.
- No dictionary or databases for text are available, except the one that is under preparation by the Urdu Language Authority but so far, their Web shows a slow progress.
- Even there is no standard keyboard, National Language Authority has concocted a keyboard in which the foremost utilized characters are set beneath the main fingers, but it is exceptionally different from the one as of now in use. Moreover, software vendors still have to adopt it as even Windows Vista is using its own version of Urdu keyboard.
- The research completed on Urdu is very tattered and is outside from the Urdu/Arab domain.
- As such no particular gatherings or meetings are arranged up until this point.
- Algorithms that are made for other language contents are not used for Urdu language.

2.4 Characteristics of Urdu Characters

This segment gives a complete rundown of attributes of the Urdu/Arabic characters to delineate the ideas. We intend to give a source to specialists to begin with. These qualities are seen from character recognition perspective.

- Urdu is one of the dialects which are composed from right to left in both printed and when written by hand.
- There aren't any upper or lower cases in Urdu Language, yet now and again the last character of a word is considered as capitalized in light of the fact that it consistently stays in its full structure.
- The state of the character shifts as indicated by its position in the word.
- Each Urdu character has either two or four particular structures. Clearly this will expand the amount of classes to be seen from.

- Urdu Language is constantly composed cursively. Urdu Words are isolated by spaces. Be that as it may, there are 6 characters that can be associated distinctly from the right, these are: wao, zey, rey, zwad, daal, alif.
- Urdu characters are 'normally' related on a nonexistent line called pattern and each letter set in a character has some fixed size dependent upon the pen ('Qalam') used which is called 'khat'.
- Character on varying zones changes its shape just as its size also.
- Some of the Urdu characters have dots which are related with the character, these specks can be above or underneath the character.
- A few characters have closed circle (suggest Table 1).Hamza (ء) crisscross shape, is definitely not an authoritative letter yet it can cause trouble in the segmentation process as it looks like with the character ein (ع) of the Urdu language.
- There are just three characters that speak to vowels و^ا or ی. In any case, there are some other shorter vowels that are spoken to by diacritics as over scores or underscores however the use of these over scores and underscores in Urdu is less as contrast with Arabic tongue. Dabs related with the character can be spoken to as two secluded spots, top, contacted specks or as a stroke.
- There is one more style of composing Urdu content which is the inventive or aesthetic calligraphy (otherwise called khatati). This style is generally loaded with covering making the recognition procedure significantly more troublesome by individual as opposed to by PCs.

2.5 Existing Solutions and their flaws:

The table below summarizes the features of available market solutions for Urdu OCR and their flaws. These are the most popular and most widely used OCR software in myriad organizations.

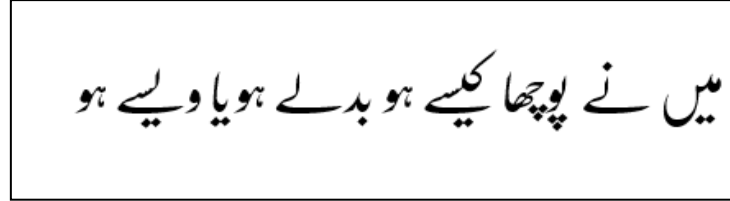


Figure 2.2: Test Image

TOOL	TEST RESULT	ANALYSIS
i2OCR	یس نے پوچھا کے ہو پر نے ہوا ولے ہو	Average
Free Online OCR	مس نے اچھا کی ہو پر نے ہو یا ولے ہو	Bad
Text Extract	No result displayed.	Poor
Urdu Nastalique OCR	Does not deal with the test image format i.e. .PNG Limitation observed.	---

Table 2.3: Existing Solutions and their flaws

2.6 Issues addressed by Urdu OCR

Traditional solutions for Urdu optical recognition have average to poor results. They have a number of limitations that the user has to deal with while using such solutions. A few of the OCR software deal with multiple image formats (with poor results) or a single image format which clearly bounds the user. Also, while testing this software, a few of them took significantly much time for processing. One of the software did not even generate a result. Some of the propriety solutions are expensive making it impossible for medium scale organizations for usage. Urdu OCR specifically targets this important issue offering a solution that is indigenous, real time, cost-effective, sports a user intuitive GUI and has an up to date knowledge base. Clients are just a few clicks away from converting their hard-printed Urdu text to editable text files, ready to be stored/ edited digitally. Urdu OCR has been developed with scalability in mind. It is suitable for layman usage, small and medium enterprises, Government institutions, Military and Tech Giants alike; truly a toolkit for all.

2.7 Novelty of Urdu OCR:

Urdu OCR has edge over most popular OCR solutions, since it has the following attributes:

- Easy and user- intuitive GUI
- Allows editing the input image at hand
- Fast Processing
- Excellent results
- Cost Effective and Indigenous Scalable Solution
- Deals with multiple image formats

All the claims are justified through the comparison between Urdu OCR and other solutions tabulated below.

Features	I2OCR	Free Online OCR	Text Extract	Urdu Nastalique	Urdu OCR
Easy GUI	✓	✓	✓	✓	✓
Image Editing	✗	✓	✗	✗	✓
Fast Processing	✓	✓	✗	✓	✓
Good Result	Satisfactory	✗	N/A	✗	✓
Multiple Formats	✓	✓	✓	✗	✓

Table 2.4: Comparison of Urdu OCR with existing solutions

2.8 Summary

Urdu language is written in the Arabic literature with an extra letter set. It acquires all the complexities of Arabic script including its cursive nature of works, right to left style of composing and change of structure and shape when a character is set at various positions in a word, loop, half shut characters and specks on above or underneath a character. National Language Authority characterized 58 characters set yet it has 41 working characters next to numeral and diacritics.

Chapter 3: Technological Requirements

3.1 Hardware Requirements

3.2 Software Requirements

3.3 Operating Systems Requirements

3.4 Setting up Working Environment

Technological Requirements

This chapter provides comprehensive details about technical requirements of Urdu OCR i.e. software, hardware and OS requirements. This chapter will provide details about the tools, python library and packages installation that have been used.

3.1 Hardware Requirements

The Hardware segments required for the development of the project include:

3.1.1 Laptop Computer

The major hardware requirement for this project is a PC which is an i5 7th Generation having 8GB RAM, 150GB hard drive and Intel® HD Graphics 630 (Kaby Lake GT2) to house Ubuntu OS and the OCR software to run smoothly (recognition of words).



Device name	Lenovo-Y520-15IKBN
Memory	7.7 GiB
Processor	Intel® Core™ i5-7300HQ CPU @ 2.50GHz × 4
Graphics	Intel® HD Graphics 630 (Kaby Lake GT2)
GNOME	3.28.2
OS type	64-bit
Disk	103.0 GB

Figure 3.1: Laptop Computer with specifications

3.2 Software Requirements

Urdu OCR's software requirements have been discussed below.

3.2.1 Ubuntu

Urdu OCR makes use of Ubuntu Operating System 18.04.4 LTS in 64-bits. This OS contains other modules required by the Urdu OCR to function properly.

3.2.2 PyCharm

PyCharm is an integrated development platform utilized in PC programming, explicitly for the Python language. The PyCharm being used is 2019.3.4 (Community Edition) with default python interpreter Python 3.6.

3.2.3 Python Programming Language

Following are the main python libraries used

PyQt5 contains the uic Python module. This module can stack .ui records powerfully making a UI. Like the uic utility; it additionally produces the Python code which will be utilized to make the UI. PyQt5's pyuic5 utility is an order line interface to the uic module.

PyQt5's QtGui includes classes for windowing framework, 2D designs, event handling, basic imaging, textual styles and text. It manages the rudimentary structure squares required for graphical UI (GUI) applications worked with Qt. Every GUI segment (names, catches and so on.) is a gadget which is put some place inside a UI window or might be shown as a different window and is a piece of QtGui.

PyQt5's QtWidgets are the essential components for making UIs in Qt. Widgets can show information and status data, get client input, and give a container to different widgets that ought to be assembled together. The Qt Widgets Module gives a lot of UI components to make desktop style UIs.

PIL (Python Imaging Library)

Python Imaging Library is a free library for the Python programming language that incorporates support for opening, controlling, and sparing a wide scope of picture designs.

These include:

- Per-pixel controls.
- Picture sifting, for example, obscuring, smoothing, or edge finding.
- Picture upgrading, for example, texture of photo, modifying brilliance, and shading.
- Adding content/words to pictures.

3.2.4 Google Tesseract

Tesseract is an OCR engine for various working systems. It is free, released under the Apache License, Version 2.0 and improvement has been supported by Google. Tesseract can process right to-left content, for example, Arabic or Hebrew and now Urdu. Tesseract is reasonable for use as a backend. Tesseract's yield will have low quality if the information pictures are not preprocessed to suit it: Images (particularly screen captures) must be scaled up with the end goal that the content x-height is at any rate 20 pixels, any turn or slant must be rectified or no content will be perceived and dull borders must be physically removed, or they will be confounded as characters.

3.2.5 Open CV (CV2)

Open Source Computer Vision Library is a library of computer programming that aims on computer vision. The Computer Vision is basically a field which utilizes the aspect of what a human eye can see and what the brain then comprehends about it. This library uses that aspect.

3.3 Operating System Requirements

The operating system used for Urdu OCR is **UBUNTU 18.04**.

This version is the best arrival of Ubuntu in years. This release of Ubuntu (LTS) focuses more on moderate programming refinement than they do on significant changes. Despite that, customarily it's not considered as an awful thing, and individuals select to ride on LTS release for the consolation of continuous help. This new Minimal Install option of Ubuntu gets the core OS, yet with less applications introduced. Ubuntu makes a Swap document rather than a Swap partition during establishment. This change won't influence the installs that are done previously and is applied on frameworks where it is required. The change assists with improving the exhibition.

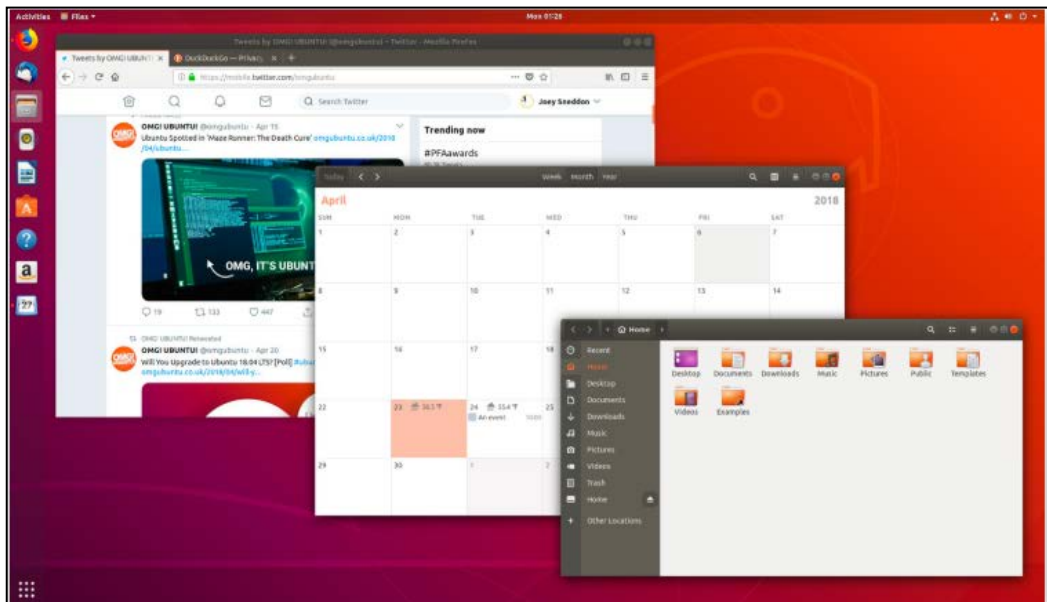


Figure 3.2: Ubuntu 18.04 LTS features

Ubuntu 18.04 System requirements

Processor Speed

It is suggested that we need a processor that requires only 1GHz, for standard installations. It is believed that this is the least at which the Ubuntu server can run. As indicated by our conclusion, at least 2GHz is required for excellent performance. This can be especially evident in the occasion that we're planning to run more resource-intensive applications.

If our equipment doesn't fulfill these requirements then there's the alternative of doing a minimal install of Ubuntu. For this Ubuntu requires at least 300MHz to run.

RAM

The least specification required for standard installations is 512MB or 1GB. Only we can use the Live Server installer on AMD64 systems.

Once more, these are the exposed least determinations. It's suggested that we should have 2GB of RAM, for any installation technique we use. This gives us a couple of headroom to run those applications for which we require greater RAM. For minimal installations, we only require 384MB of RAM. But such a little RAM is not acceptable so we won't recommend it.

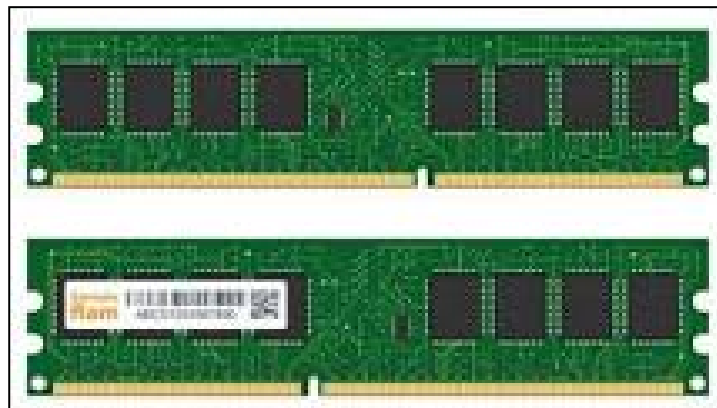


Figure 3.3: Random Access Memory (RAM)

Disk Space

It's prescribed to utilize 1.5GB for the base establishment in case of Disk space. But for extra facilities and features we need 2.5GB of disk space. In real, we'll be needing considerably more disk space than this. So, this won't be a big deal. The issue is that how much space is required for the installation on our drive. For minimal installation, 1.5GB of disk space is required for the base system and 2.5GB for the full installation.

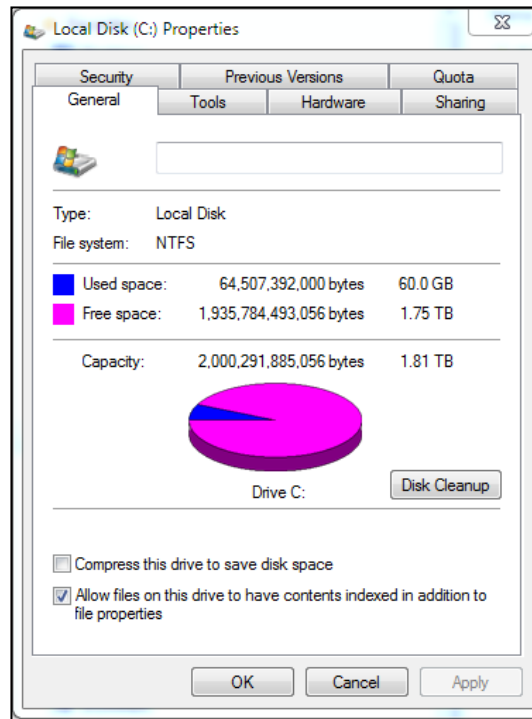


Figure 3.4: Disk Properties

Other Requirements

Our other requirements include a graphics card. We'll be needing a monitor too with a resolution of 640x480, USB port, a CD drive and also internet for the installation. It must be noted that Ubuntu Server Edition only supports AMD64, ARM, POWER8, z Systems, and Linux ONE. Long Term Server has advanced equipment and techniques. It's because of the inclusion of the **new Kernel**.

Linux Kernel 4.15

We have Linux kernel 4.15 at the core of Ubuntu 18.04.

Linux 4.15 helps the latest peripheral devices, graphics cards and USB ports to work “out of the box” and exceptionally good, and also helps them improve other devices performances. Moreover, the Linux 4.15 based Ubuntu also provides advanced power management for systems with SATA Link Power Management, provides encrypted and secured memory management on AMD hardware, Linux security module stacking support — and much more.

3.4 Setting up Working Environment

Our project is based on the Ubuntu version 18.04. Ubuntu 18.04 LTS iso file is downloaded from its official site. Following stage is to copy the downloaded ISO picture into the USB/DVD or flash drive to boot the PC from that drive. Once the machine is powered up, we select our preferred installation i.e. graphical install.

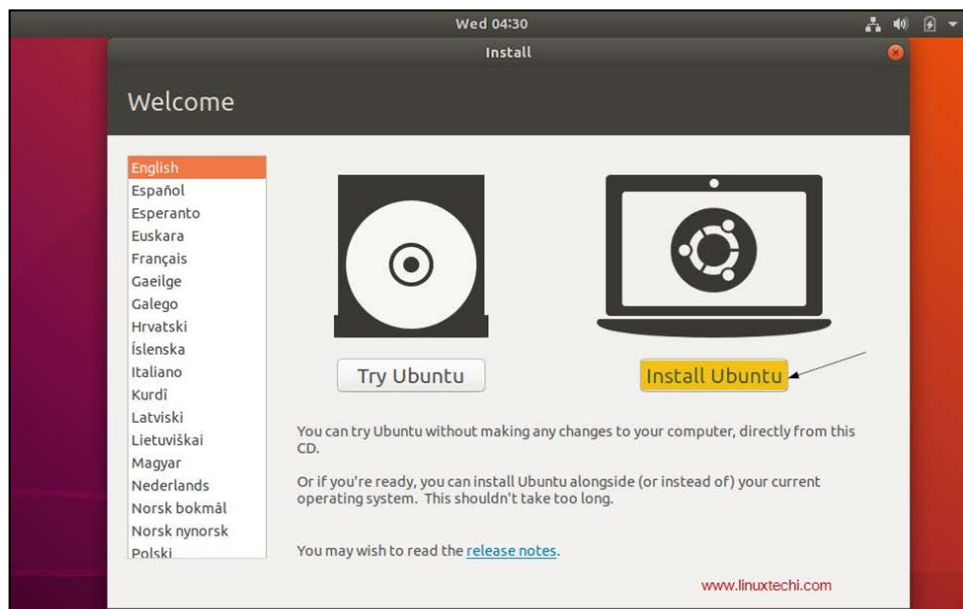


Figure 3.5: Ubuntu Linux installation menu

Next couple of screens will prompt to select information such as preferred keyboard layout and type of installation.

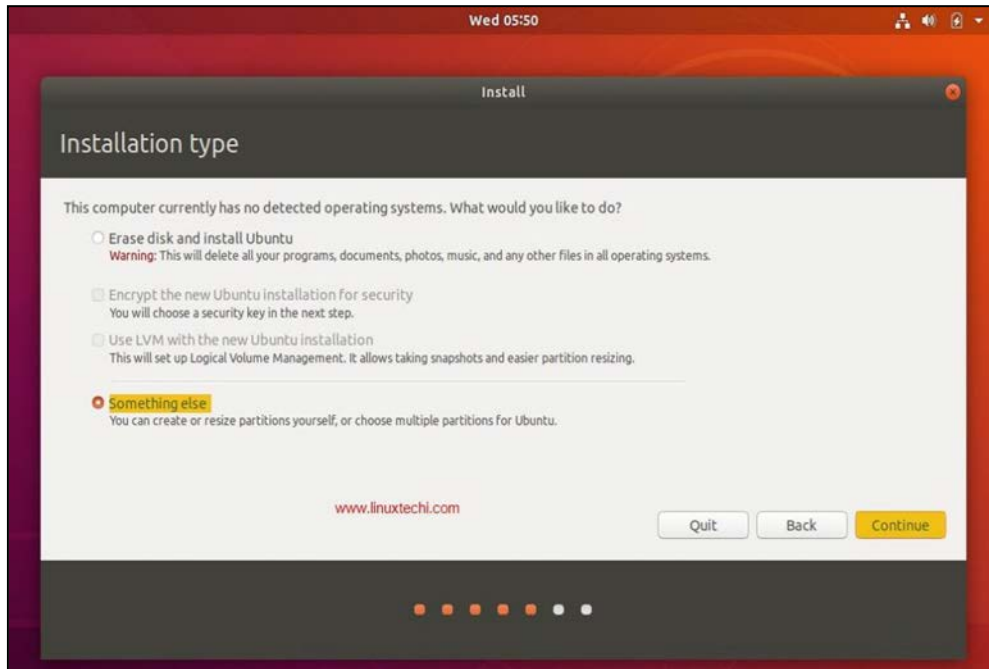


Figure 3.6: Custom partition option selection

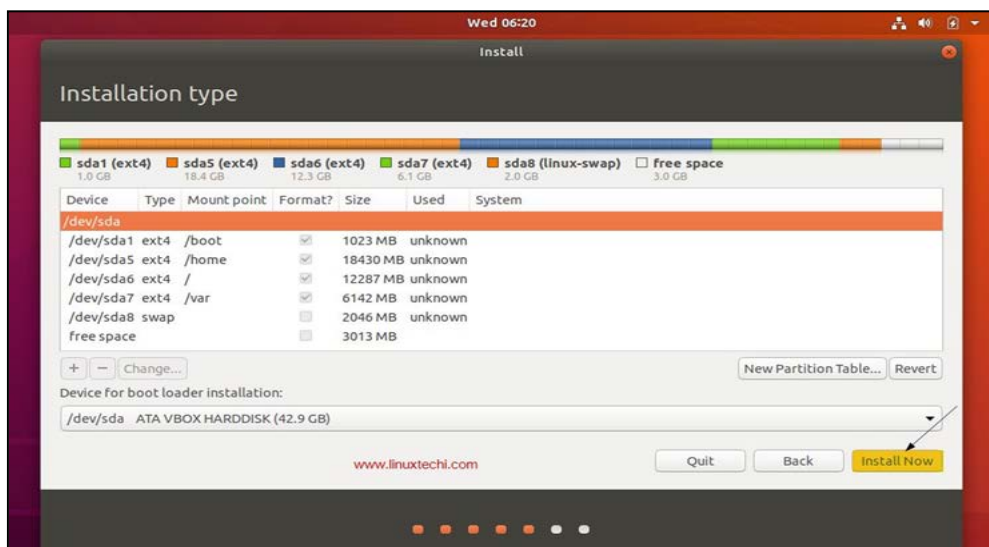


Figure 3.7: Partition creation task complete

After the partitions are created as per our needs, we proceed the installation by clicking the Install Now button.

After that, time zone is selected, and we provide our necessary credentials for log in.

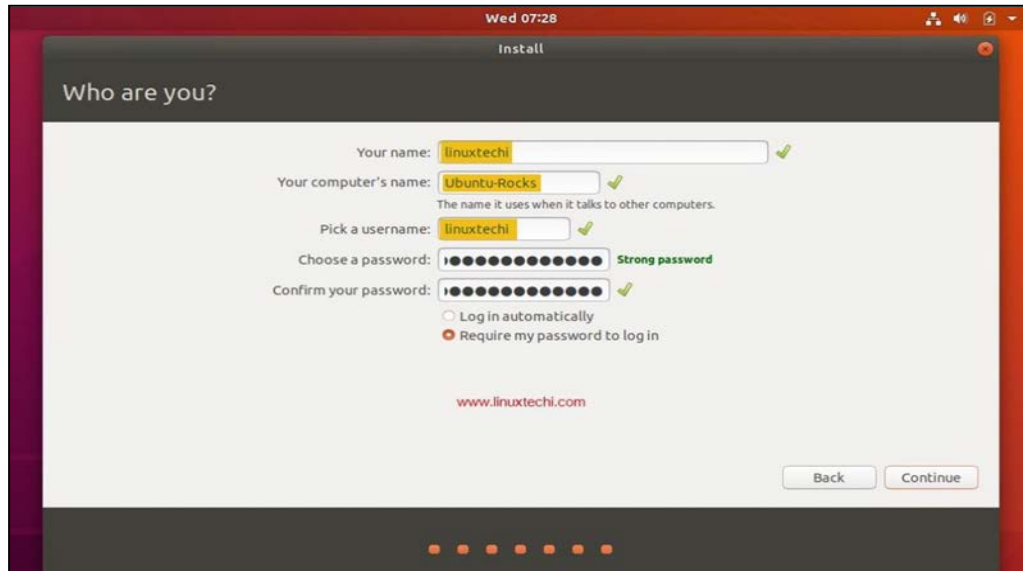


Figure 3.8: Credentials put up

The process of Ubuntu 18.04 LTS begins now and takes around 5-10 mins. When the process is finished, USB/DVD is expelled from the drive and we click Restart Now to restart our machine.

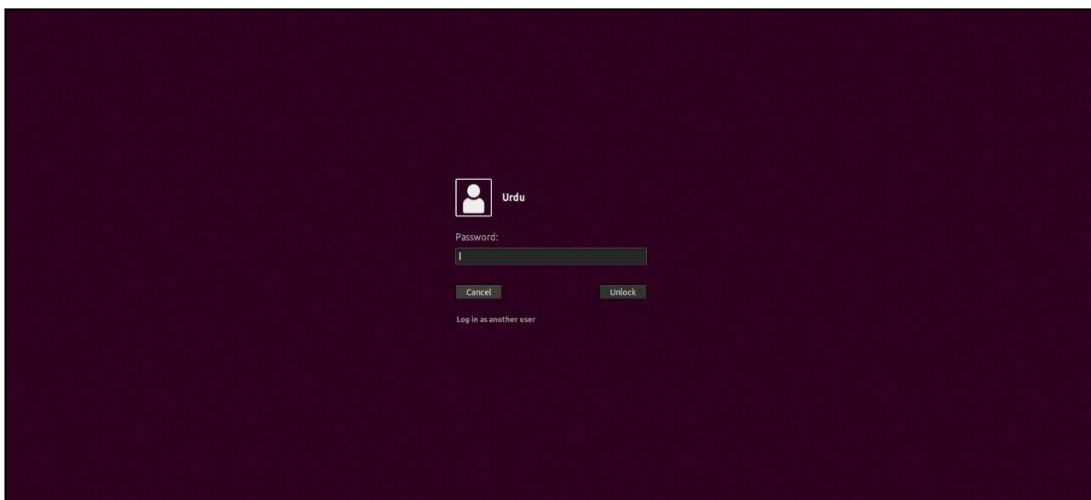


Figure 3.9: Log in password

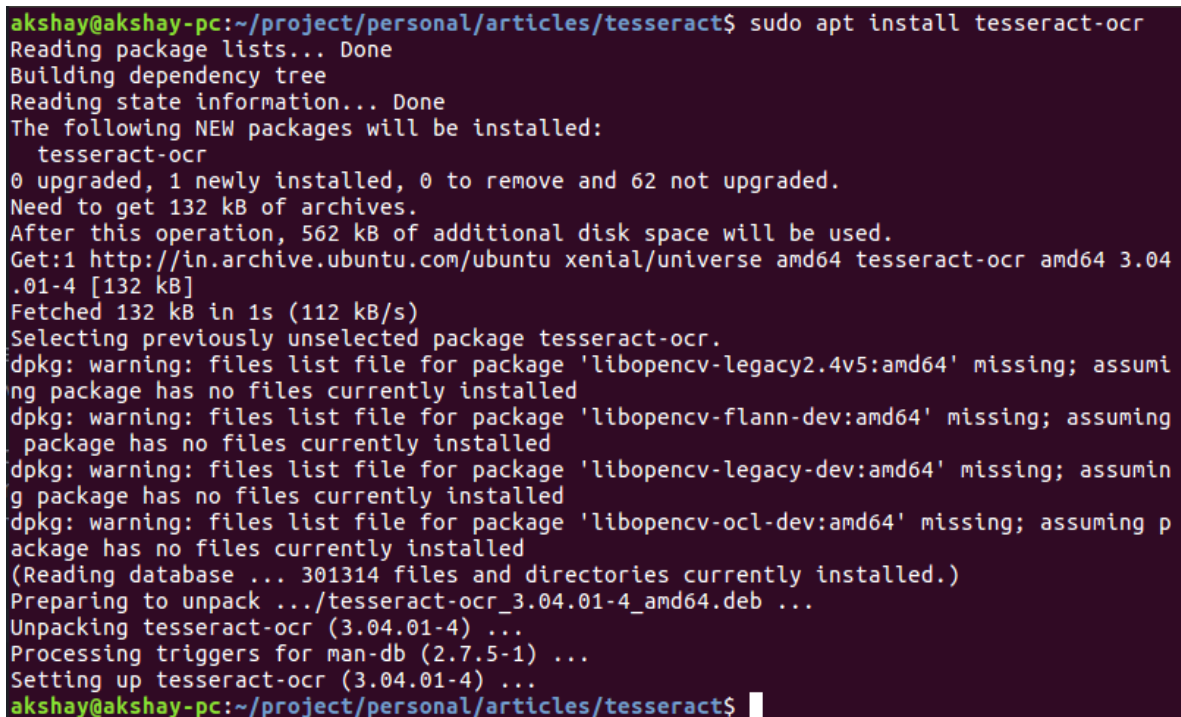
After the installation of Ubuntu, we move towards downloading the PyCharm platform for python.

The following command is written on the command line.

```
Sudo snap install pycharm-community --classic
```

It is pretty simple to install tesseract; the following commands are run on the command prompt:

```
Sudo apt update  
sudo apt install tesseract-ocr  
Sudo apt install libtesseract-dev  
Sudo pip install pytesseract|
```

A terminal window screenshot showing the command 'sudo apt install tesseract-ocr' being executed. The output shows the package being installed, the disk space requirements, and the successful completion of the installation. The terminal prompt is 'akshay@akshay-pc:~/project/personal/articles/tesseract\$'.

```
akshay@akshay-pc:~/project/personal/articles/tesseract$ sudo apt install tesseract-ocr  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  tesseract-ocr  
0 upgraded, 1 newly installed, 0 to remove and 62 not upgraded.  
Need to get 132 kB of archives.  
After this operation, 562 kB of additional disk space will be used.  
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 tesseract-ocr amd64 3.04  
.01-4 [132 kB]  
Fetched 132 kB in 1s (112 kB/s)  
Selecting previously unselected package tesseract-ocr.  
dpkg: warning: files list file for package 'libopencv-legacy2.4v5:amd64' missing; assumi  
ng package has no files currently installed  
dpkg: warning: files list file for package 'libopencv-flann-dev:amd64' missing; assumi  
ng package has no files currently installed  
dpkg: warning: files list file for package 'libopencv-legacy-dev:amd64' missing; assumi  
ng package has no files currently installed  
dpkg: warning: files list file for package 'libopencv-ocl-dev:amd64' missing; assumi  
ng package has no files currently installed  
(Reading database ... 301314 files and directories currently installed.)  
Preparing to unpack ../tesseract-ocr_3.04.01-4_amd64.deb ...  
Unpacking tesseract-ocr (3.04.01-4) ...  
Processing triggers for man-db (2.7.5-1) ...  
Setting up tesseract-ocr (3.04.01-4) ...  
akshay@akshay-pc:~/project/personal/articles/tesseract$
```

Figure 3.10: Successful Tesseract installation

The next step is the installation of the Tesseract lingual packages.

```
Sudo apt-get install tesseract-ocr-[lang]
```

3.4.1. Installing the Necessary Libraries

The three most important libraries for our project are the PyQt5, PIL and cv2 libraries. Following commands are used to install the above-mentioned libraries:

For installing PyQt5, we first install the pip library.

```
sudo apt update
sudo apt install python3-pip
```

Afterwards, typing the commands below in the terminal ensures successful installation of PyQt5.

```
Pip3 install --user pyqt5
sudo apt-get install python3-pyqt5
sudo apt-get install pyqt5-dev-tools
sudo apt-get install qttools5-dev-tools
```

For installing PIL, we first have to do the following working:

```
$ sudo apt-get install python-dev libjpeg-dev libfreetype6-dev zlib1g-dev
$ Sudo ln -s /usr/lib/`uname -i`-linux-gnu/libfreetype.so /usr/lib/
$ sudo ln -s /usr/lib/`uname -i`-linux-gnu/libjpeg.so /usr/lib/
$ sudo ln -s /usr/lib/`uname -i`-linux-gnu/libz.so /usr/lib/
```

By now we are aptly able to install PIL.

```
$ sudo pip install pil
```

In order to integrate OpenCV from Ubuntu 18.04 repositories, following steps are carried out:

The packages index is refreshed and the OpenCV package is installed by typing:

```
sudo apt update
sudo apt install python3-opencv
```

Chapter 4: Methodology

4.1 Tesseract Training

4.2 Image Pre-processing

4.3 Recognition

4.4 File Creation

Methodology

This chapter describes a comprehensive methodology of Urdu OCR which explains in detail the complete cycle from training the model to the creation of the editable text file for the input image.

4.1 Tesseract Training

Tesseract 4.00 incorporates a replacement neural system-based recognition engine that gives fundamentally higher precision (on document images) than the previous renditions, reciprocally for a critical increment in required computation power. It is configured as a text line recognizer with its origins in the Python Long Short-Term Memory network implementation. As for running Tesseract 4.0.0, it's useful but not essential to possess a multi-core (4 is good) machine, with OpenMP and Intel Intrinsic support for SSE/AVX extensions. As Tesseract has an inbuilt urd.traineddatafile (which has a character error rate of 76% and word error rate of 75%), we used the Fine Tune model for training. Fine tuning is the process of training an existing model on new data without changing any part of the network. As the first step, we build Tesseract and Leptonica; Leptonica being the software that contains the library for image processing and image analysis.

After the buildup, we create a Temp directory with the name 'tesstutorials' where all the training data will be saved. We then create a 'langdata' directory and load all the necessary langdata files from the internet sources. The radical_stroke.txt, common.punc, font_properties, Arabic.unicharset and Arabic.xheights files are the necessary files.

Afterwards, we create an 'urd' directory where the urd files will be loaded (GitHub source). The necessary urd files for training Urdu language are:

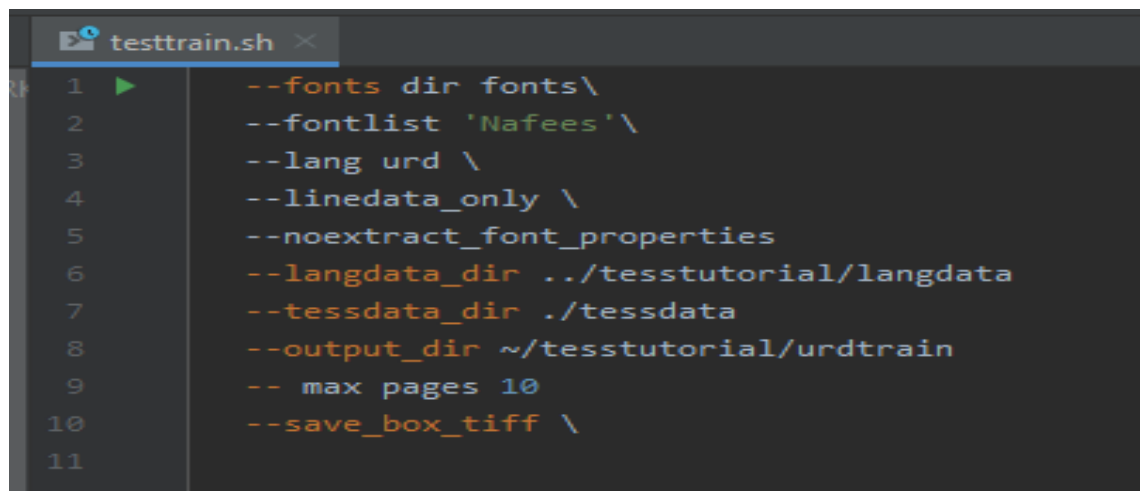
urd.training_text, urd.punc, urd.numbers, urd.wordlist, desired_characters, forbidden_characters, urd.singles_text, urd.unicharset and okfonts.txt.

After that, we download the best urd trained data file that is the already trained Urdu file available in the Tesseract plug-ins.

The next step is to create our own training data set that is done through the tesstrain.sh script. We download the font file from the internet which is in the .ttf format. In our case, it's the Nafees Naksh font. For training, Tesseract first needs a box file to go with each picture. The box file may be a document that lists the characters within the training image, in order, one per line or with the coordinates of the bounding box round the image.

Following is the testtrain.sh file created for lstm training:

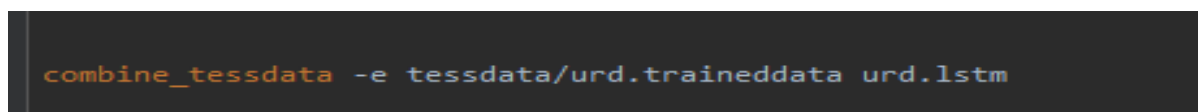
(The necessary files cloned through GitHub)



```
testtrain.sh x
1  --fonts dir fonts\
2  --fontlist 'Nafees'\
3  --lang urd \
4  --linedata_only \
5  --noextract_font_properties
6  --langdata_dir ../tesstutorial/langdata
7  --tessdata_dir ./tessdata
8  --output_dir ~/tesstutorial/urdtrain
9  -- max pages 10
10 --save_box_tiff \
11
```

Figure 4.1: Python tesstrain.sh script for creating training data

A recognition model is often extracted from an existing traineddata file, using combine_tessdata. As the next step, we extract the LSTM model from the urd.traineddata file.



```
combine_tessdata -e tessdata/urd.traineddata urd.lstm
```

Figure 4.2: Python code for extracting LSTM

Then we have to evaluate the model on the new font i.e. Naskh. It is done through the following python script.

```
lstmeval
--model urd.lstm
--traineddata tessdata/urd.traineddata
--eval_listfile ../tesstutorial/urdtrain/urd.training_files.txt
```

Figure 4.3: Python script for evaluating lstm

Next step is to fine tune the lstm model on the new font ‘Naskh/Nafees’. Using the pre-trained urd.traineddata file, we add our own training data under the urd.training_files.txt for adapting the software for our font and for more accuracy.

```
--continue_from urd.lstm
--model_output ../tesstutorial/output/nafees
--traineddata tessdata/urd.traineddata
--train_listfile ../tesstutorial/urdtrain/urd.training_files.txt
--max_iterations 4000
```

Figure 4.4: Python script for Fine tuning LSTM

We also declare checkpoints so as a means for fault tolerance in cases where the network / system connection gets disrupted during training which goes as follows.

```
--stop_training --continue_from ../tesstutorial/output/nafees_checkpoint --traineddata tessdata/urd.traineddata --model_output ../tesstutorial/output/nafees.traineddata
```

This marks the end of the training process.

After evaluation of the final LSTM model, following results are received i.e. a character error rate of only 2.9% and a word error rate of mere 11.6%. Recognition results are shown in the later part of this paper.

```
Nafees Naskh:
```

```
Base Model:
```

```
At iteration 0, stage 0, Eval Char error rate=85.279819, Word error rate=80.07428
```

```
Fine-tunned Model:
```

```
At iteration 0, stage 0, Eval Char error rate=2.9307867, Word error rate=11.609509|
```

Figure 4.5: Evaluations results

4.2 Image Preprocessing

Pre-processing is the name for work done with pictures at the most reduced level. The purpose of this is an improvement of the image data that covers for instance pivoting, scaling, translation etc. The steps to be taken are:

- Upload image
- Resize image/Crop
- Rotation/Skew Correction

Uploading an image takes an image from your Personal Computer to be processed by the OCR. The image must be in .jpg, .png, .jpeg or .bmp image format to be read by the OCR otherwise the image would be unreadable.

```
self.extract_button.clicked.connect(self.extract_button_click)
def select_button_click(self):
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    file_name, _ = QFileDialog.getOpenFileName(self, "QFileDialog.getOpenFileName()",
    """
        "Image Files (*.png)", options=options)
    if file_name:
        pixmap = QPixmap(file_name)
        self.image_label.file_name = file_name
        self.image_label.setText("")
        self.image_label.setPixmap(pixmap)
```

Figure 4.6 Python command for Uploading Image

There is the resize or crop feature which removes the background noise or useless interference in the image and focuses on the text more. As the OCR requires a minimum of 200-300 DPI to work successfully. Resizing the image greatly increases the chance of successful character recognition.

```
self.crop_image_button.setGeometry(420, 380, 120, 40)
self.crop_image_button.setText('Crop')
```

Figure 4.7 Geometry for Crop Feature

```
def crop_image_button_click(self):
    self.crop_image_window = CropImageWindow(self, self.image_label.pixmap())
    self.crop_image_window.show()
```

Figure 4.8 Python command for Crop Feature

The next edit feature is the rotation correction. The user can manually input the no. of degrees for which the image needs to be rotated. A Skewed picture is a picture which isn't straight so, slanted pictures legitimately sway the line division of OCR engine which lessens its exactness.

```
def rotate_image_button_click(self):
    rotate_angle = self.rotate_degree_textEdit.toPlainText()

    try:
        rotate_angle = int(rotate_angle)

        transform = QTransform()
        transform.rotate(rotate_angle)
```

Figures 4.9 Python command for Rotation

4.3 Recognition

This is the main and most important module of OCR system. Appropriate features from segments or units are extracted and are then passed to the recognition system for identification. Generally, recognition system is trained beforehand for that data. So, we have also used our trained data set to recognize the image text. Our Urdu OCR allows the user to click on the extract button after uploading the image and extracts the text from the image.

```
def extract_button_click(self):
    if self.image_label.file_name:
        self.hide()
        self.image_text_window = ImageTextWindow(self, self.image_label.pixmap())
        self.image_text_window.show()
    else:
        QMessageBox.question(self, 'URDU OCR message', "Select an Image", QMessageBox.Ok)
```

Figure 4.10 Extract Text Function

Here, **self** represents the instance of the class. By using the “self” keyword we can bind the attributes with the given arguments.

Main Window is the class which allows user to select, edit or extract the text from the image. When we click the extract button, this extract function takes place which is shown in the image. As depicted in the image, we are calling the class `ImageTextWindow` because the extracted text is shown to the user using this class.

We used tesseract OCR engine for which we had to import **tesseractocr** which integrates directly with Tesseract's C++ API using Python that allows for a simple Pythonic and easy-to-read source code. While processing an image in tesseract, it enables real concurrent execution when used with Python's threading module by releasing the GIL. So using **tesseractocr.image_to_text** we convert the contents of the image into our desired text.

```
import tesseractocr
from PIL import Image
import os
class OCR:
    path = os.getcwd() + "/ocr/tessdata"
    @staticmethod
    def convert_to_text(img_path):
        img = Image.open(img_path)
        return tesseractocr.image_to_text(img, lang="urd", path=OCR.path)
```

Figure 4.11 OCR command for Image to text Conversion

4.4 File Creation

This is the last step of Urdu OCR. Once the text is recognized and extracted from the image, it can be saved into a text file so that it can be used for later purposes i.e. to be used for some other application, copying it, to be emailed etc. So, after clicking the save button, it connects it with the `save_button_click` function.

```
self.save_button.clicked.connect(self.save_button_click)
```

Figure 4.12 OCR command for saving file

We have a class **ImageTextWindow** which allows the user to save that text through creating a text file.

```
self.parent().show()
def save_button_click(self):
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    fileName, _ = QFileDialog.getSaveFileName(self, "QFileDialog.getSaveFileName()", "",
                                              "Text Files (*.txt)", options=options)
    if fileName:
        print(fileName, _)
        text_file = open(fileName, "w")
        text_file.write(self.output_editText.toPlainText())
        text_file.close()
```

Figure 4.13 Directory for Saving

The **QFileDialog** is an inbuilt class which provides a dialog that allow users to select files or directories.

Options argument holds various options that tells us how to run the dialog. **GetSaveFileName** is a static function that will open a dialog box to select a file name. As we have to save the text file so we have filtered it too. In this way, the user will be able to save the text file in his/her directory.

Chapter 5: Results & Analysis

5.1 Final Deliverable

5.2 Image Editing Results

5.3 Recognition Results

5.4 File Creation

5 Results and Analysis

This chapter summarizes the results and outcomes of Urdu OCR.

5.1 Final Deliverable



Figure 5.1: Final Deliverable

The final deliverable of the project is a desktop based software application that performs the extraction of Urdu text from images and pastes the data into an editable text file, all through the clicks of a few buttons.

The main GUI page of Urdu OCR asks the user to select an image with Urdu text in it. The extraction afterwards, is only a few clicks away. The input image can be any of the following formats:

.jpeg, .png, .jpg and .bmp.

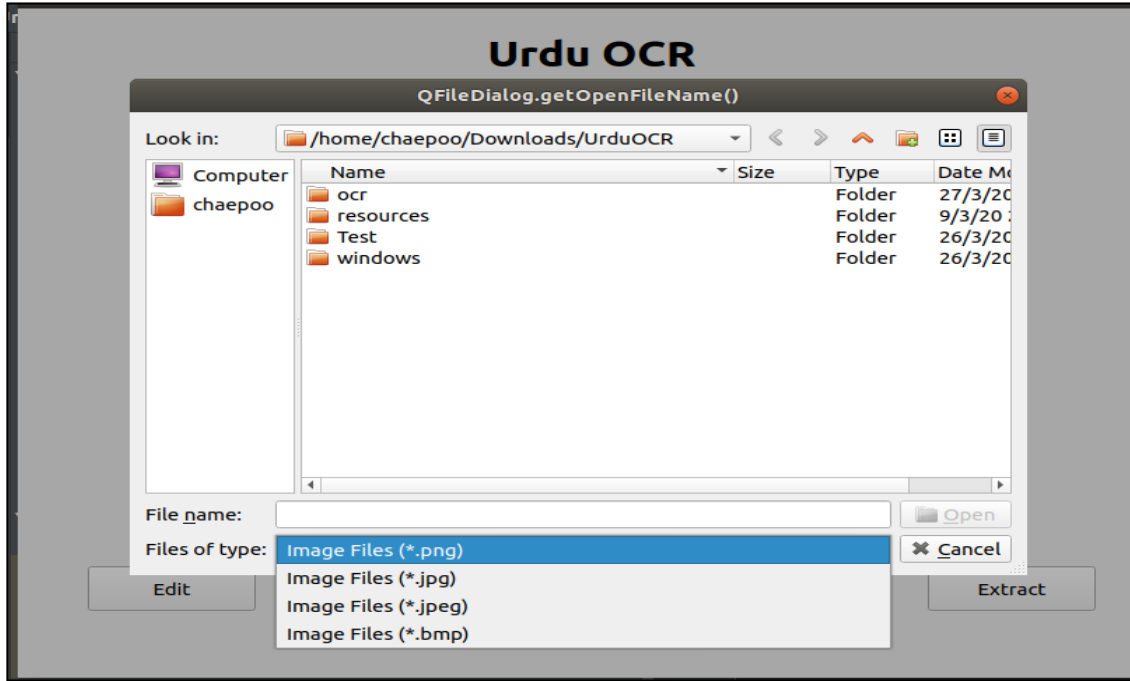


Figure 5.2: Select Image Dialogue Box

5.2 Image Editing Results

The main page of the application asks the user to select an image from his/her device's directory. The image may be skewed or slanted at any angle possible, but that can be edited at real time by clicking the 'Rotate' button. Rotate button works through rotating the image by degrees. The angle of rotation is inputted by the user so as to straightly align the input text image.



Figure 5.3: Edit Image Screen

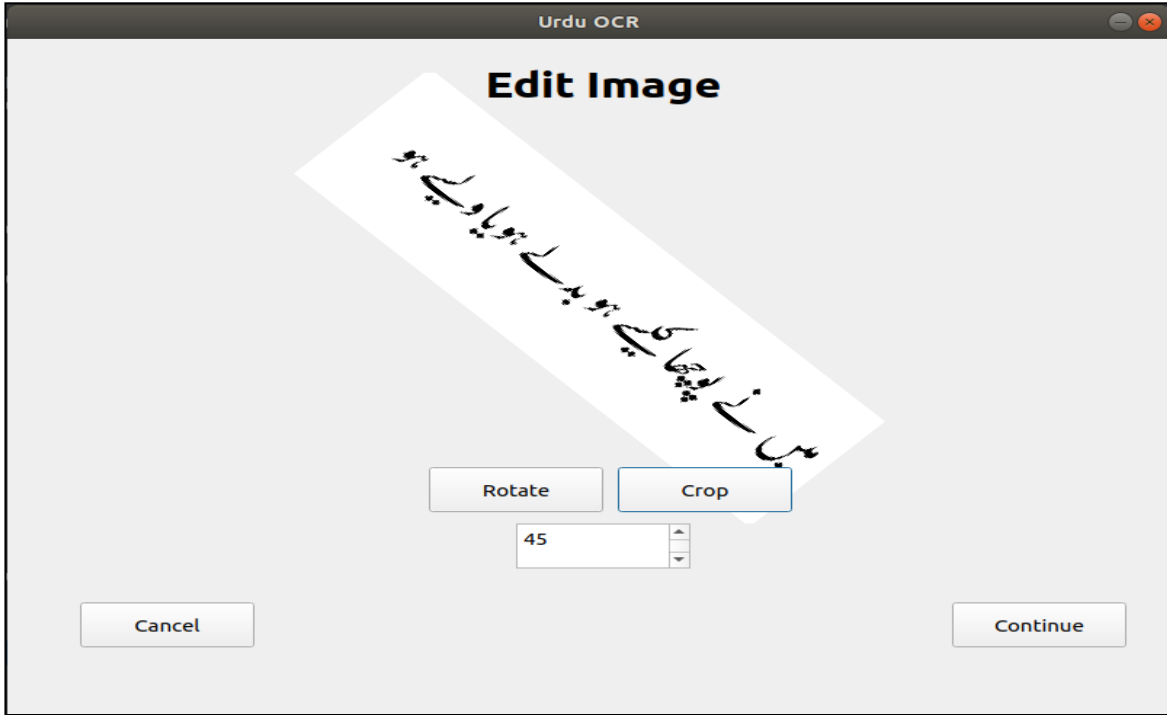


Figure 5.4: Rotating Image with Degrees

The user can also crop the image if the user wills to extract only a portion of the input image. That is done by simply clicking the 'Crop' button which prompts the user to crop the image through the rectangular borders.

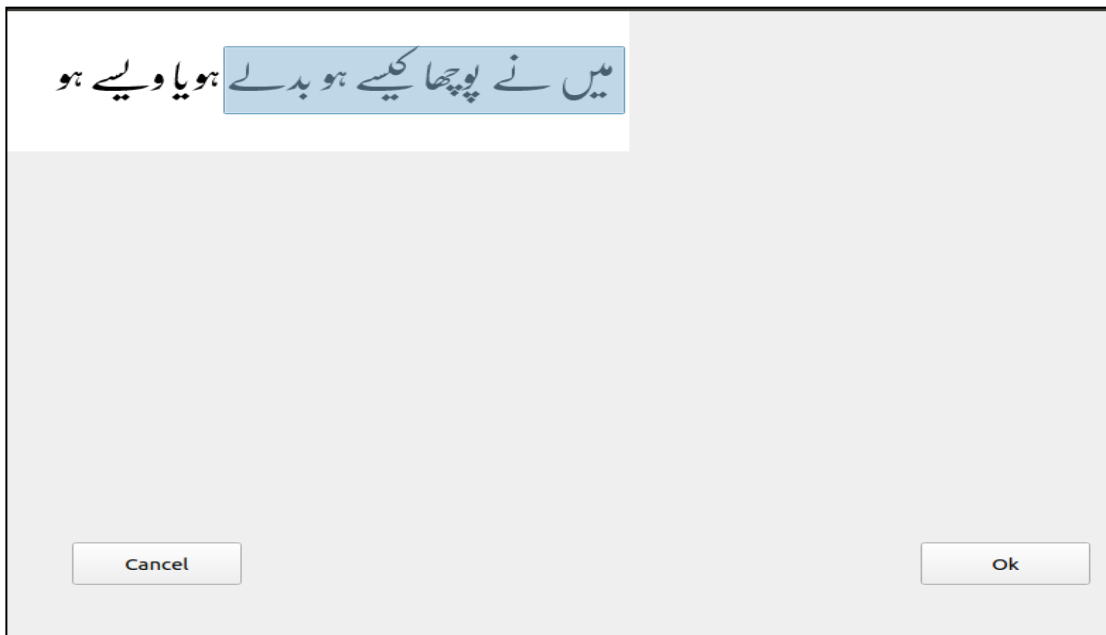


Figure 5.5: Cropping the input Image

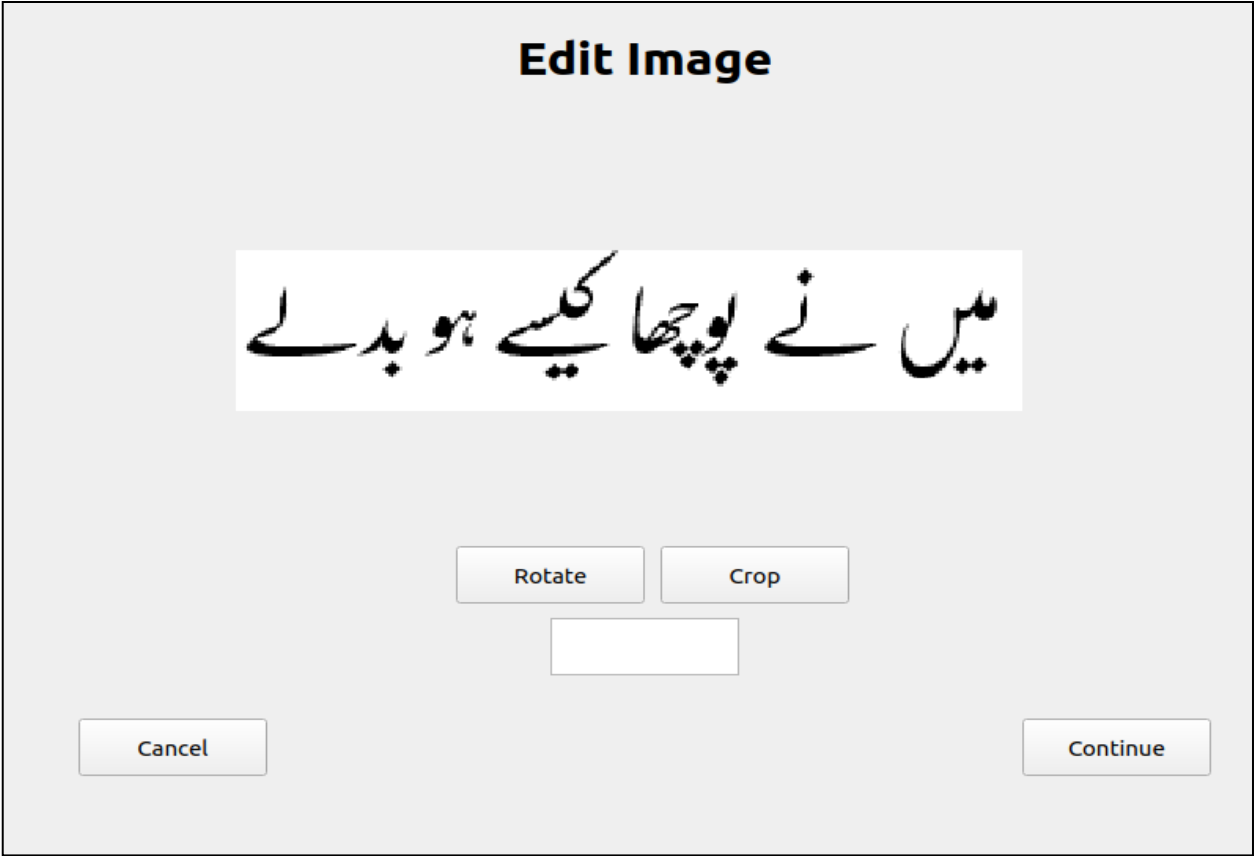


Figure 5.6: After cropping out desired portion of the image

5.3 Recognition Results

When the input image is selected and edited by the user as per his/her needs, the user then clicks the 'Continue' button to initialize the recognition process. After fast processing, the extracted text is displayed to the user in a rectangular text box.

The recognition results of Urdu OCR are summarized below.

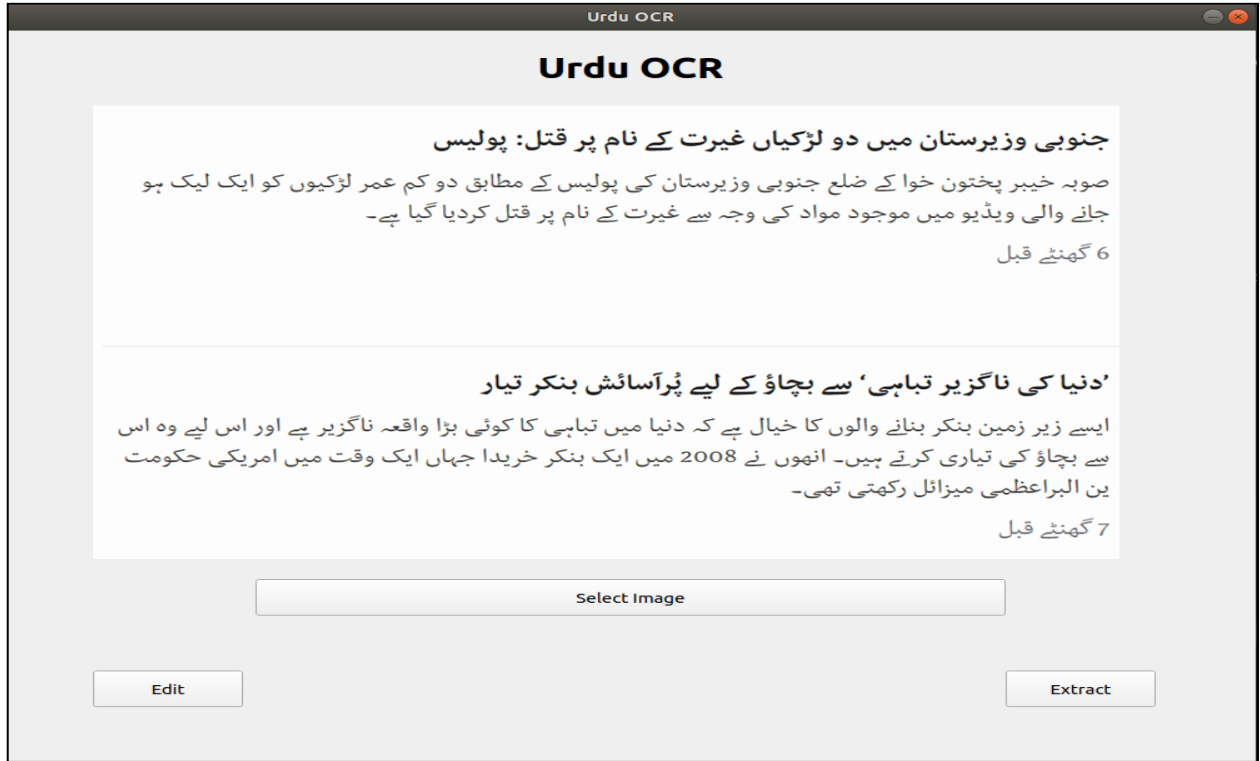


Figure 5.7: A multi-line test image in Naskh font

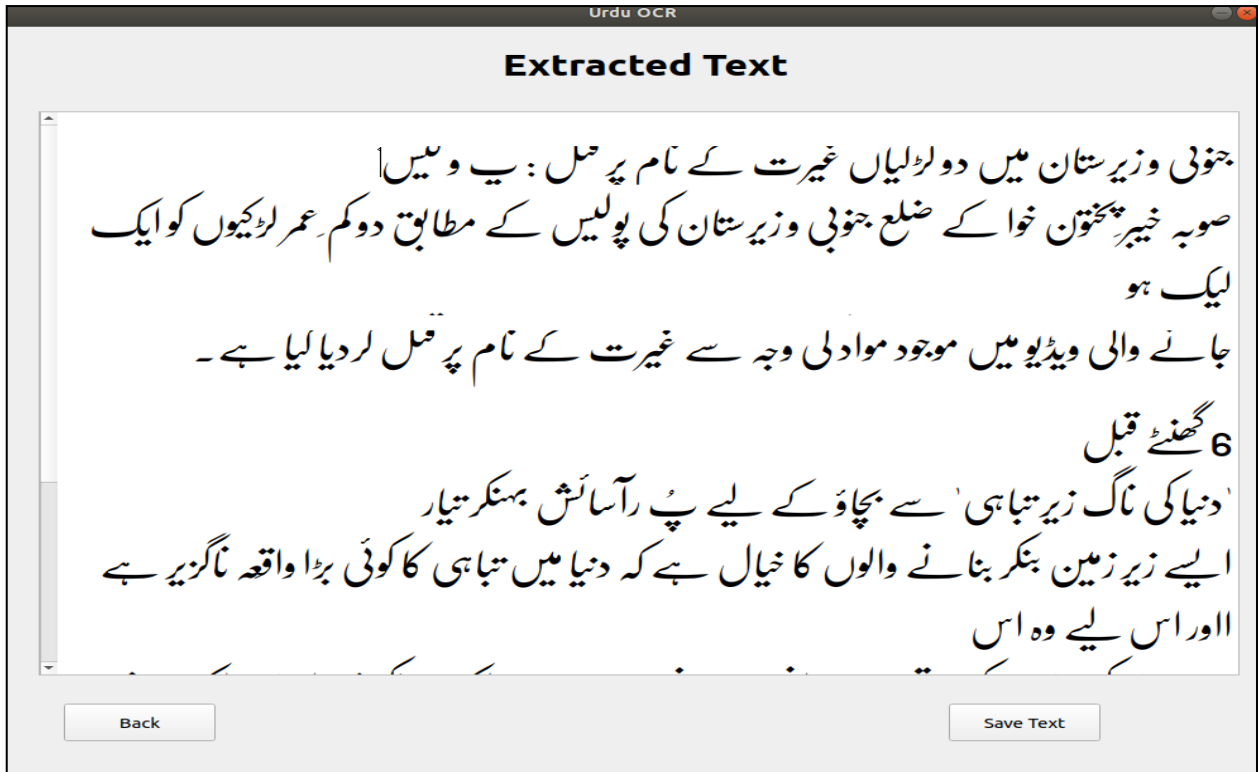


Figure 5.8: Recognition results

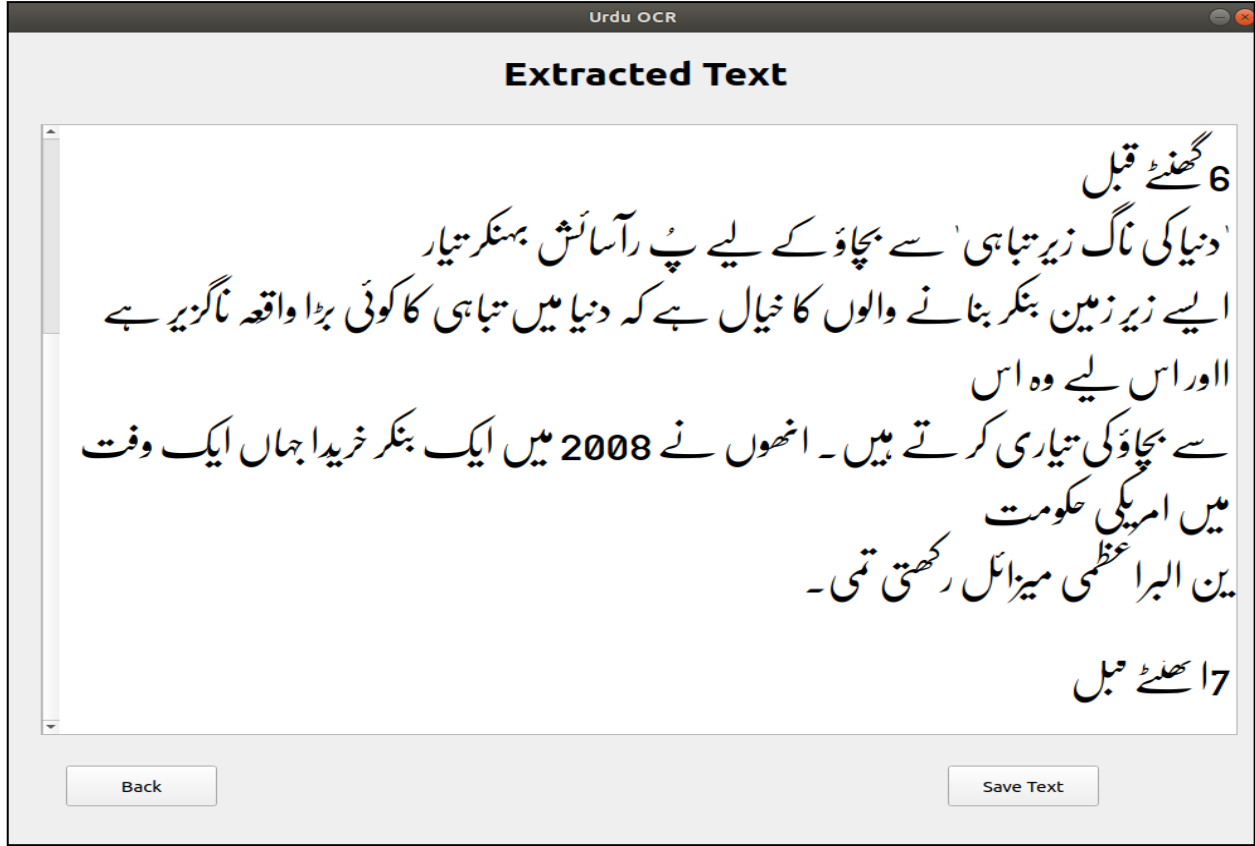


Figure 5.9: Recognition results (continued)

As it is evident from the extracted results, 96% success rate is achieved. A few words are not extracted correctly but again, in this case the characters are extracted accurately, the error being that the characters are not linked together in the best manner. Moreover, our software can recognize numbers with ease as well, as it is evident from the recognition results attached above. Speaking of the font sizes that can be recognized perfectly range from size 16 and above.

During testing, we observed that our OCR software can also recognize Urdu text written in the Anjali Old Lipi font, which is also written along the horizontal line, as Naskh.

Following images provide the extraction results for the Urdu text in the font Anjali Old Lipi.



Figure 5.10: Anjali Old Lipi Urdu Text Test Image

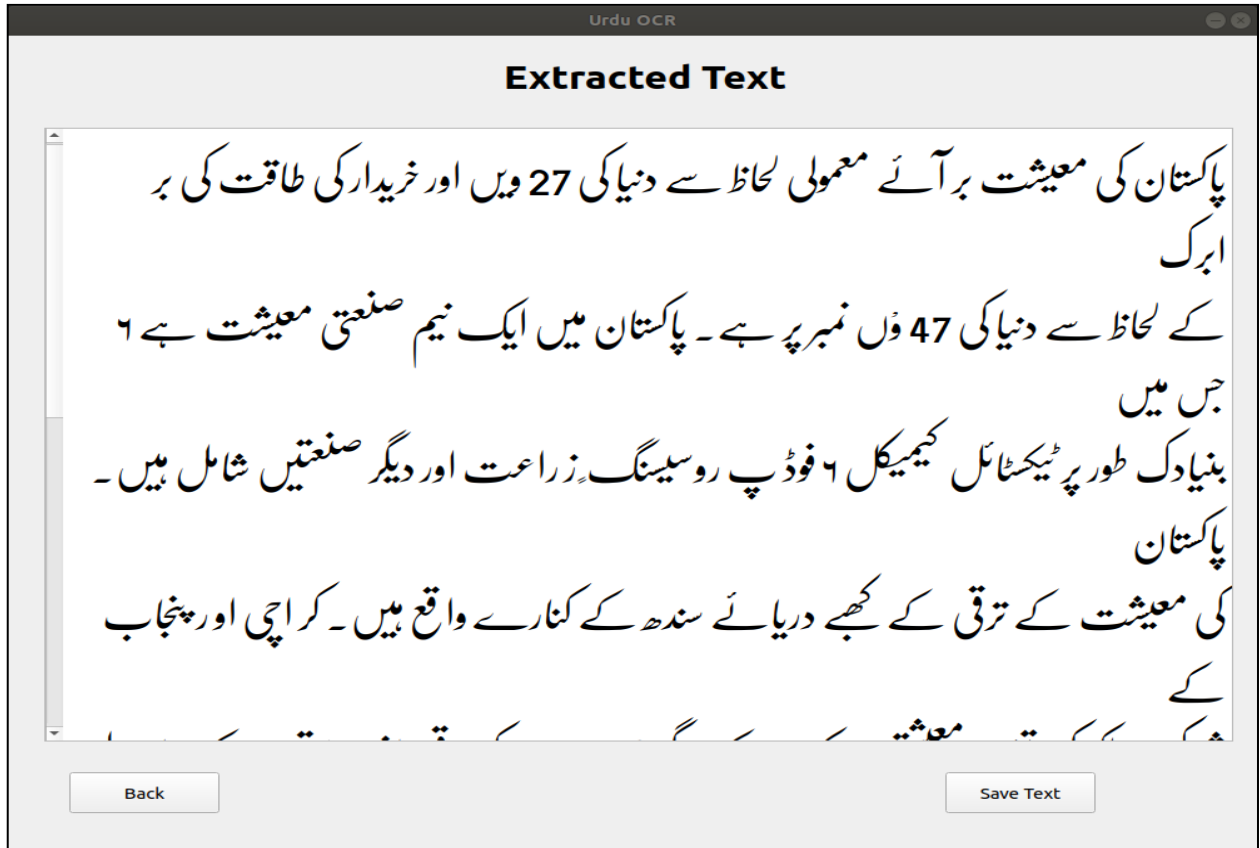


Figure 5.11: Recognition result

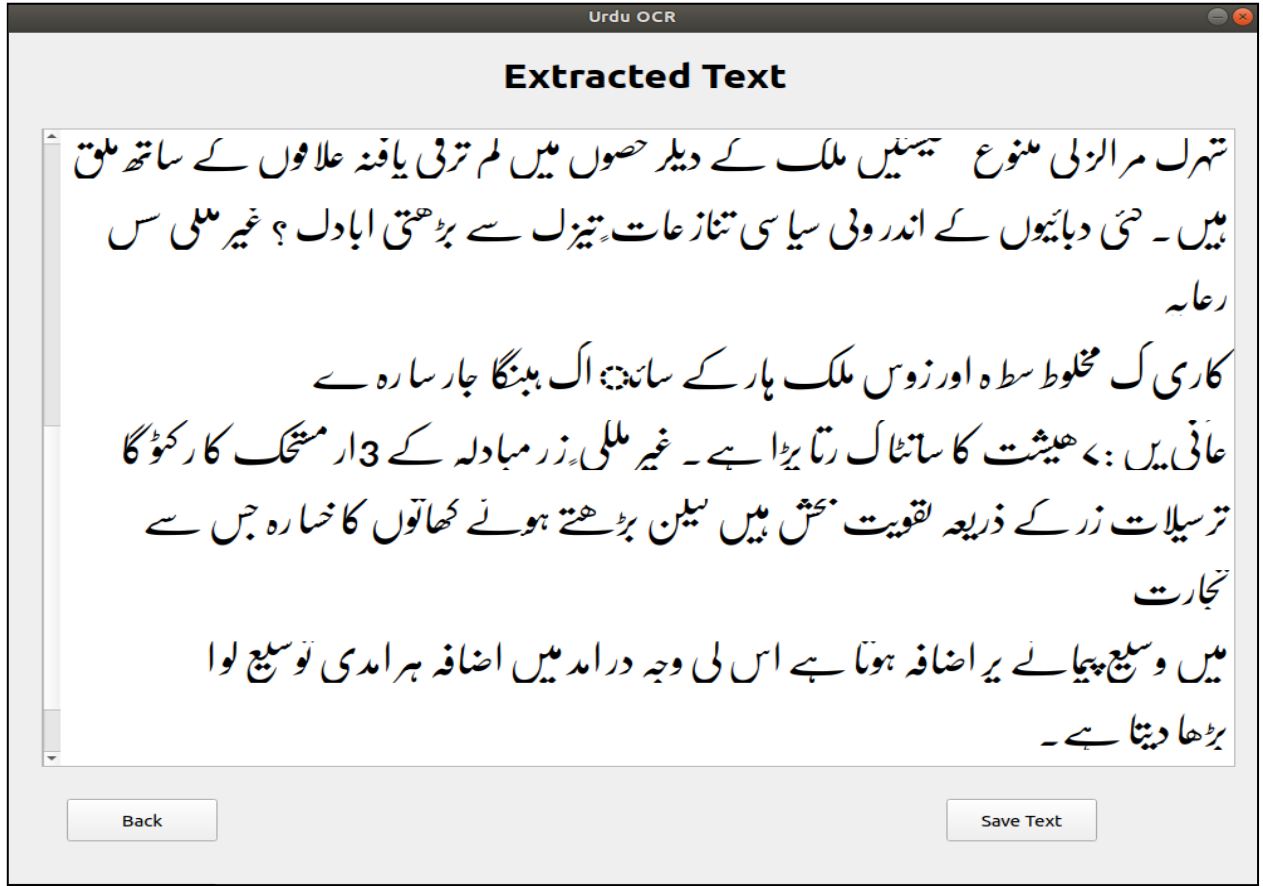


Figure 5.12: Recognition result (continued)

5.4 File Creation

After the extraction process, the recognized text is displayed to the user. It's up to the user to either cancel after viewing the results (to select another image) or save the text onto an editable text file with the .txt format. It's done through clicking the 'Back' and 'Save Text' buttons respectively. When the user clicks the 'Save Text' button, the application prompts for the file name to save the file as well as the location where the file will be saved. File creation marks the end of the process.

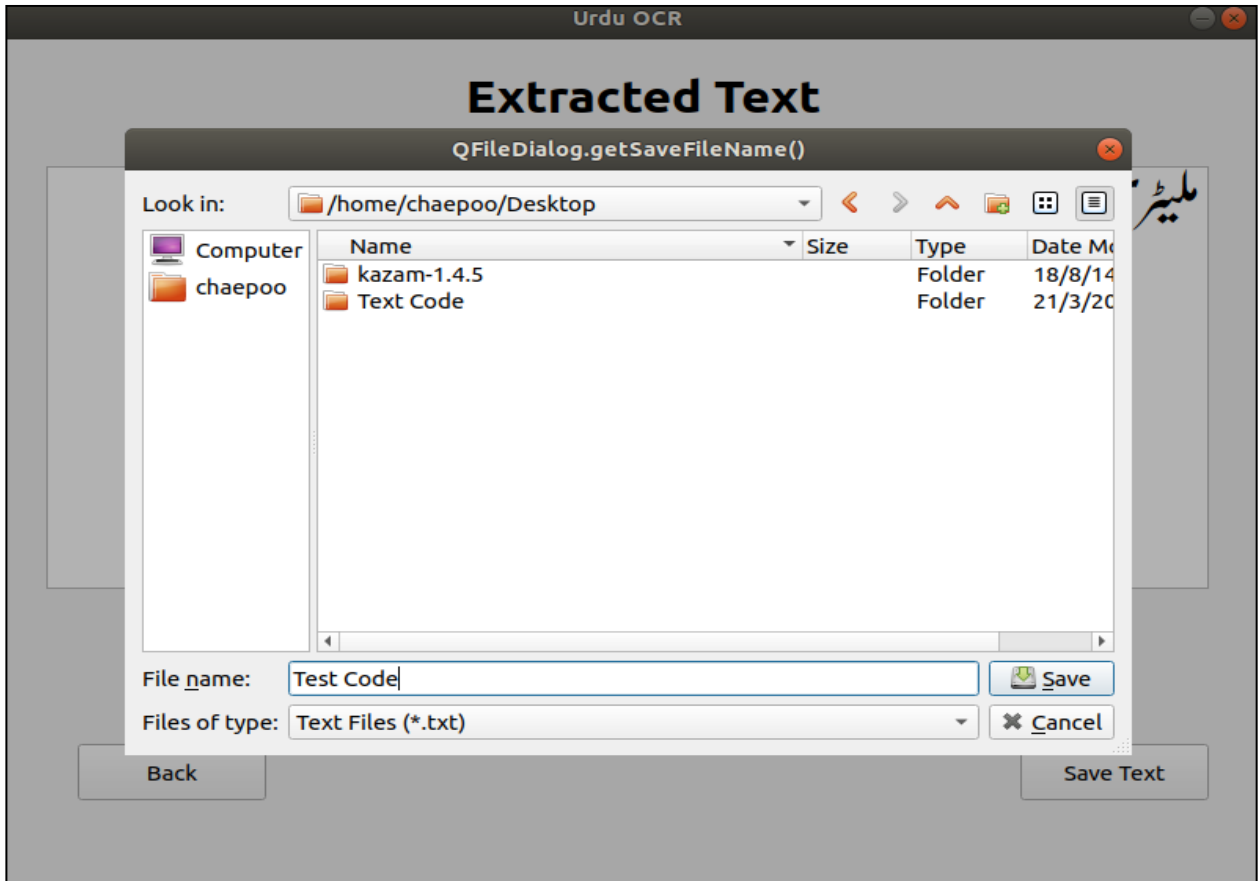


Figure 5.13: Creating .txt file dialogue box

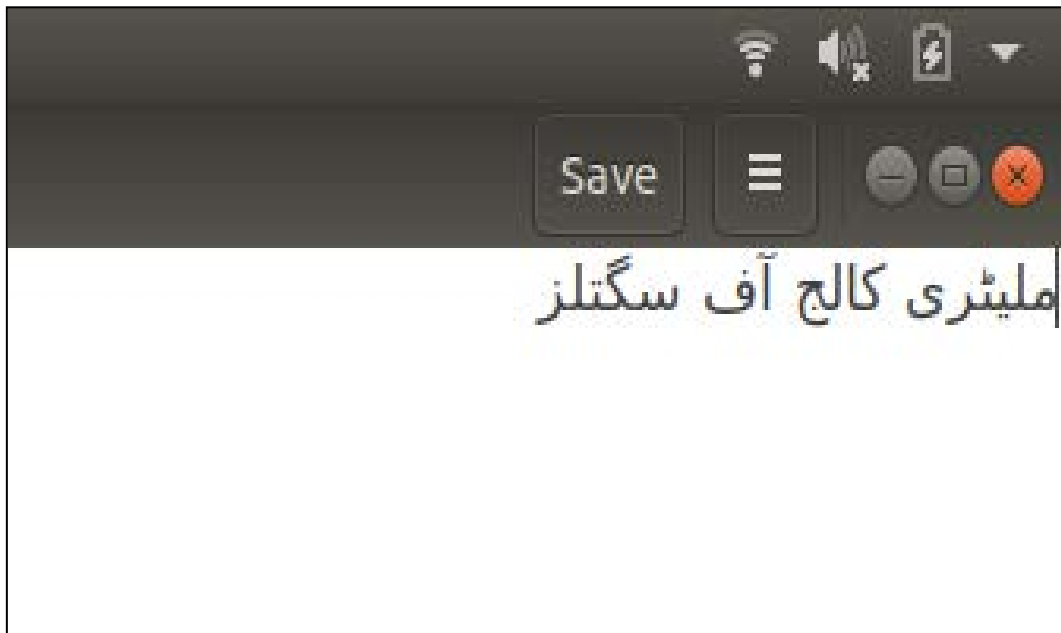


Figure 5.14: .txt file saved in the user selected directory

Chapter 6: Conclusion & Future Work

6.1 *Conclusion*

6.2 *Future Work*

Conclusion and Future Work

6.1. Future Work

Urdu OCR can be extended in several directions. Some of them are given below:

- Development of multiple fonts so that other than Naskh, Urdu OCR can work with different fonts too.
- Urdu OCR will be able to work with different font sizes.
- Development of hand-written OCR system. In this way Urdu OCR will be able to extract handwritten text as there is huge demand for such feature.
- It can be evolved into a bigger and more complex system with more features and functionality.
- Text to voice conversion can also be added to this application to assist the visually impaired.
- A database can be maintained to help user keep a record of his performance throughout.
- Noise removal and Skew Correction can also be added which will help to process real data.
- In future, Urdu OCR will be able to deal with missing words as sometimes strokes are absent in between characters and so it becomes difficult to recognize the word.

6.2 Conclusion

OCR is one of the most captivating and challenging areas of pattern recognition. It can contribute a lot to the advancement of an automation process. It can also improve the interface between man and machine in many applications. In this paper, we presented a review of OCR work done on Urdu language. Here, we briefly discussed different steps used in OCR development and work done on Urdu text recognition. This can be used as a starting point to develop the OCR system for Urdu text. During the review of OCR systems, we have identified that there are no complete and accurate OCR systems because of more similarity of characters and compound characters. Hence, a detailed font study can help in finding good solutions for these challenges.

References

- [1] [Muaz, Ahmed] “Urdu Optical Character Recognition System.” *Cle.org.pk*, 2010, [Online]
www.cle.org.pk/Publication/theses/2010/OCRMUAZ.pdf.

- [2] [Shabbir, Safia & Siddiqi, Imran] (2016). Optical Character Recognition System for Urdu Words in Nastaliq Font. *International Journal of Advanced Computer Science and Applications*. [Online]

- [3] [Zaheer Ahmed] (2008) “Urdu Optical Character Recognition using Neural Networks.” *Scribd.com*, [Online]
<https://www.scribd.com/doc/63785963/Urdu-Optical-Character-Recognition-OCR-Thesis-Zaheer-Ahmad-Peshawar-Its-Source-Code-is-Available-on-MATLAB-Site-21-01-09>

- [4] [Adrian Rosebrock] (2018) “OpenCV OCR and text recognition with Tesseract” *Pyimagesearch.com*, [Online]
<https://www.pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract/>

- [5] [Online], Available:
https://raw.githubusercontent.com/tesseractocr/langdata_lstm/master/radical-stroke.txt

- [6] [Online], Available:
https://github.com/tesseract-ocr/tessdata_best/raw/master/urd.traineddata

- [7] [Pradeep Kumar] (2020) “Ubuntu 18.04 LTS Desktop Installation Guide with Screenshots”. *linuxtechi.com*, [Online]
<https://www.linuxtechi.com/ubuntu-18-04-lts-desktop-installation-guide-screenshots/>

- [8] [Dr Sarmad Hussain] (2012) “Urdu Nastalique Optical Character Recognition System”
<https://ignite.org.pk/urdu-nastalique-optical-character-recognition-system/>
- [9] [Mark Lahn] (2018) “Ubuntu System Requirements”
<https://www.servermania.com/kb/articles/what-are-the-requirements-for-ubuntu-server/>
- [10] [Safia Shabir, Imran Siddique] (2016) “ Optical Character Recognition System for Urdu Words”
https://www.researchgate.net/publication/303868982_Optical_Character_Recognition_System_for_Urdu_Words_in_Nastaliq_Font
- [11] [Brijesh Gupta] (2018) “Improve Accuracy of OCR using Image Preprocessing”
<https://medium.com/cashify-engineering/improve-accuracy-of-ocr-using-image-preprocessing-8df29ec3a033>
- [12] [Kang & Atul] (2019) “Optical Character Recognition Pipeline: Image Preprocessing”
<https://theailearner.com/2019/05/29/optical-character-recognition-pipeline-image-preprocessing/>
- [13] [Zaki, Urooba & Hakro, Dil & Khoumbati, Khalil-Ur-Rehman & Zaki, Muhammad & Hameed, Maryam] (2019).” Issues & Challenges in OCR”
https://www.researchgate.net/publication/333117928_Issues_Challenges_in_Urdu_OCR/citation/download
- [14] [Online], Available:
https://raw.githubusercontent.com/tesseractocr/langdata_lstm/master/radical-stroke.txt

- [15] [Saeeda Naz, Khizar Hayat, Muhammad Imran Razzak, Muhammad Waqas Anwar] (2013) “The Optical Character Recognition of Urdu-like Cursive Scripts”
<https://www.sciencedirect.com/science/article/abs/pii/S0031320313004135>
- [16] [Ng Wai Foong] (2019) “A Beginner’s Guide to Tesseract OCR”
medium.com, [Online]
<https://medium.com/better-programming/beginners-guide-to-tesseract-ocr-using-python-10ecbb426c3d>
- [17] [Herbert F. Schantz] (2007) "Urdu Nastaleeq Optical Character Recognition"
<https://www.semanticscholar.org/paper/Urdu-Nastaleeq-Optical-Character-Recognition-Ahmad-Orakzai/882a9f21fdf94cab16fc2ed7b8d7d6b867cda499>
- [18] [Zaheer Ahmed, Jehanzeb Khan Orakzai] (2019) “A Beginner’s Guide to Tesseract OCR” *medium.com*, [Online]
<https://medium.com/better-programming/beginners-guide-to-tesseract-ocr-using-python-10ecbb426c3d>
- [19] [Naila Habib Khan, Awais Adnan] (2020) “Urdu Optical Character Recognition Systems: Present Contributions and Future Directions”
<https://ieeexplore.ieee.org/document/8438450>
- [20] Akram, Qurat ul Ain & Hussain, Sarmad & Niazi, Aneeta & Anjum, Umair & Irfan, Faheem. (2014). Adapting Tesseract for Complex Scripts: An Example for Urdu Nastalique. 191-195. 10.1109/DAS.2014.45. 2019)

- [21] Tabassam, Nawaz & Naqvi, Syed & Rehman, Habib & Anoshia, Faiz. (2009). Optical Character Recognition System for Urdu (Naskh Font) Using Pattern Matching Technique. *International Journal of Image Processing*. 3.
- [22] [Joey Sneddon] (2019) “Ubuntu 18.04 LTS: What’s New and where to download”? [Online], Available:
<https://www.omgubuntu.co.uk/2018/04/ubuntu-18-04-download-release-features>
- [23] [Z. Ahmad, J. K. Orakzai, I. Shamsheer and A. Adnan] “Urdu Nastaleeq Optical Character Recognition, *World Academy of Science, Engineering and Technology* 1(8), 2380–2383, 2007
<https://doi.org/10.1.1.193.3286>.
- [24] [S. S. R. Rizvi, A. Sagheer, K. Adnaan and A. Muhammad] (2019) “Optical Character Recognition System for Nastalique Urdu-Like Scrip Languages Using Supervised Learning” [Online], Available:
<https://doi.org/10.1.1.193.3286>.
- [25] [Adrian Rosebrock] (2018) “OpenCV Tutorial: A Guide to Learn OpenCV” *PyimageSearch.com*, [Online], Available:
<https://www.pyimageSearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/>

Appendix A: Synopsis Form – Urdu OCR

Extended Title: Optical Character Recognition (OCR) for Urdu Literature.
Brief Description of the Project/Thesis with Salient Space: OCR is a widespread technique aimed at recognizing text in images. It converts handwritten, typed or printed text into data that can be edited on a computer. We are proposing of developing an OCR software for Urdu literature particularly in Nastaliq font that would convert Urdu text in images into an editable text file. Machine-readable text can then be decoded by screen readers, tools that use speech synthesizers to read out the words on a screen so blind and visually impaired people can understand them.
Scope of Work: Our project will mainly focus on the following objective: We would develop the Optical Character Recognition for Urdu literature in 1 font.
Academic Objective: Through our proposed project, we will be able to: <ol style="list-style-type: none">1. Develop an OCR software.2. Achieve human like intelligence and reading capability in our project.3. Implement Digital Image Processing and Machine Learning.
Application /End Goal Objective: Our project is aimed at: <ol style="list-style-type: none">1. Digitizing paper documents.2. Automated Data Entry.3. Assisting the visually impaired.4. Saving time of manually entering printed data onto a computer.5. Saving space as reams of paper documents will be converted to CDs of archived documents.
Previous Work Done on The Subject: Our project is the first of its kind in our University. Similar work has been done in another university.
Material Resources Required: Hardware: <ul style="list-style-type: none">• Camera / Mobile Phone• Book stand• A book with literature in Nastaliq font Software: <ul style="list-style-type: none">• OCR processing software• Android application

No of Students Required:

Two team members are required to work on this project.

Special Skills Required:

Following skills are required to successfully develop this project:

1. Android Development
2. Machine Learning
3. Deep Learning
4. Digital Image Processing

Approved Status**Syndicate Member Names with their Signatures**

1. Ayesha Tahir Khan Khattak (Leader)
2. Manahil Ahmed (Member)
3. Muhammad Ismaeel Moeen Qadir Khan (Member)

Supervisor Name & Signature: Sir Bilal Rauf

HoD Signature _____

R&D SC Record Status

File # _____ Coordinator Signature _____

Appendix B: List of Abbreviations

OCR	Optical Character Recognition
CV2	Computer Vision
GUI	Graphical User Interface
PIL	Python Imaging Library
LSTM	Long Short Term Memory
BMP	Bit Map Image File

Appendix C: Plagiarism Report

Thesis OCR			
ORIGINALITY REPORT			
12%	8%	2%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	Submitted to Higher Education Commission Pakistan Student Paper		3%
2	www.cle.org.pk Internet Source		1%
3	pixuate.com Internet Source		1%
4	www.omgubuntu.co.uk Internet Source		1%
5	www.linuxtechi.com Internet Source		<1%
6	github.com Internet Source		<1%
7	Submitted to Heriot-Watt University Student Paper		<1%
8	link.springer.com Internet Source		<1%
9	code.google.com		

Internet Source

<1%

10

Submitted to Indira Gandhi Delhi Technical University for Women

Student Paper

<1%

11

U. Pal, B.B. Chaudhuri. "Indian script character recognition: a survey", Pattern Recognition, 2004

Publication

<1%

12

markklaver.com

Internet Source

<1%

13

A.M. Zeki. "The Segmentation Problem in Arabic Character Recognition The State Of The Art", 2005 International Conference on Information and Communication Technologies, 2005

Publication

<1%

14

www.scribd.com

Internet Source

<1%

15

Submitted to Durban University of Technology

Student Paper

<1%

16

Submitted to University of Surrey Roehampton

Student Paper

<1%

17

www.powells.com

Internet Source

<1%

Submitted to Asia Pacific Institute of

18	Information Technology Student Paper	<1 %
19	Submitted to Queen's University of Belfast Student Paper	<1 %
20	vuir.vu.edu.au Internet Source	<1 %
21	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %
22	hdl.handle.net Internet Source	<1 %
23	scholar.lib.vt.edu Internet Source	<1 %
24	Submitted to University of East London Student Paper	<1 %
25	Submitted to Cranfield University Student Paper	<1 %
26	Submitted to University of Bath Student Paper	<1 %
27	Submitted to Curtin University of Technology Student Paper	<1 %
28	Submitted to University of Sydney Student Paper	<1 %
	Submitted to University of Glasgow	