PEHCHAN

(URDU OPTICAL CHARACTER RECOGNITION)

FINAL YEAR PROJECT

SESSION: 2003 – 07

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY, UNIVERSITY OF SARGODHA
PAKISTAN

PEHCHAN

(URDU OPTICAL CHARACTER RECOGNITION)

FINAL YEAR PROJECT

SESSION: 2003 – 07

A final year project report submitted by

MUHAMMAD UMAIR ANJUM (BCS03-07R)

in partial fulfillment of the requirements for the degree of
Bachelor of Sciences of Computer Science

DEPARTMENT OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY, UNIVERSITY OF SARGODHA
PAKISTAN

DECLARATION

We declare that this is our own work and has not been submitted in any form for another degree or diploma at any university or institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Signatures:

_____ _____ _____ _____

Project Team:

Muhammad Umair Anjum

Zill-e-Subhan

Imran Ali

Muhammad Kamran

Date:

___ /___ /___.

SUPERVISOR'S APPROVAL (BS FINAL YEAR PROJECT)

Project Title:

PEHCHAN (Urdu Optical Character Recognition)

Document Type:

Final Project Report

Project Team:

Muhammad Umair Anjum	BCS03-07R
Zille Subhan	BCS03-35R
Imran Ali	BCS03-05R
Muhammad Kamran	BCS03-02R

Supervisor Comments

_____

_____

_____

Supervisors Name: Mr Fahad Maqbool

Signature: _____

Date: _____

DEDICATIONS

Dedicated to our beloved parents and respected teachers.

ACKNOWLEDGEMENTS

We here by acknowledge our prestigious supervisor with whom effort the accomplishment of our Project Report for the final year project would have been a mist. We acknowledge their help with great honor and we believe that this great personal will continue his supervision in future.

Table of Contents

Table of figures

Chapter 1: Introduction

Optical Character Recognition (OCR) is a type of document image analysis where a scanned digital image that contains either machine printed or handwritten script is input into an OCR software engine and translating it into an editable machine readable digital text format (like ASCII text).

When we scan a sheet of paper we reformat it from tangible "hard object" to a "digital object", which we save as an image. The image can be manipulated as a whole but its text cannot be manipulated separately. In order to be able to do so, we need to "tell" the computer to recognize the text. The OCR application does that; it recognizes the characters and makes the text editable and searchable, which is what we need. [2]

The subject of character recognition has been receiving considerable attention in recent years due to the advancement of the automation process. Automatic character recognition improves the interaction between man and machine in many applications like office automation, cheque verification, mail sorting, and a large variety of banking, business and data entry applications. [1]

OCR works by first pre-processing the digital image into its smallest component parts with layout analysis to find text blocks, sentence/line blocks, word blocks and character blocks. The character blocks are then further broken down into component parts, pattern recognized and compared to the OCR engines large dictionary of characters from various fonts and languages. Once a likely match is made then this is recorded and a set of characters in the word block are recognized until all likely characters have been found for the word block. The word is then compared to the OCR engine's large dictionary of complete words that exist for that language.

There are two types of OCR systems. Online OCR can get image directly from scanner and convert it into editable form while an Offline OCR get image from already maintained database of images.

Offline character recognition is the task of determining what letters or words are present in digital image of handwritten text. It is of significant benefit of man machine communication and can assist in the automatic processing of handwritten documents. [4]

Pehchan is an Offline Urdu OCR System, a graduating project for computer science department of University of Sargodha. It gets image from image repository or from any location of hard disk. After performing different operations on the image, it converts it into editable form.

1.1 Current Progress in the field

A lot of work has been done in the field of optical character recognition. Different optical character recognition systems have been developed for different languages like English, Latin, and Japanese etc. A very little amount of work has been done for Urdu and other similar languages like Arabic and Persian.

Because Urdu is a cursive language, so it is very difficult to develop a system which accurately classify and recognize the Urdu characters. Although some work has been done in the field of Urdu optical character recognition but still there is no system which accurately done segmentation of Urdu characters.

We will briefly describe some of the important work that has been done for optical character recognition of Urdu and other similar languages.

U.Pal and Anirban Sarkar of Indian Statistical Institute have presented an approach for the recognition of printed Urdu characters. They proposed an approach in which they try to recognize individual characters of Urdu using a combination of topological, contour and water reservoir concepts. [10]

S. Chandana and U. Pal proposed an approach in which they present methods for optical character recognition of different languages including English, Urdu and Dewangari. They have proposed methods for the recognition of mixed characters of different languages in same file. [7]

M.Salmani Jeloder, M. J. Fadaeieslam, N. Mozayani and M.Fazeli proposed a Persian OCR system using morphological operators. In order to recognize Persian characters they used morphological operators, especially Hit/Miss operator to descript each sub-word and using a template matching approach they have tried to classify generated description. They used just one font in two different sizes. [5]

Muhammad S. Khorsheed and William F Clocksin presented a technique for extracting structural features from cursive Arabic script. After preprocessing, the skeleton of the binary word image is decomposed into a number of segments in a certain order. Each segment is transformed into a feature vector. The target features are the curvature of the segment, its length relative to other segment lengths of the same word, the position of the segment relative to the centroid of the skeleton and detailed description of curved segments. The result of this method is used to train the Hidden Markov Model to perform the recognition. [6]

Dr. John Cowell and Dr. Fiaz Hussain proposed an approach for thinning the Arabic characters for feature extraction. In their approach they proposed a method for thinning and also discuss algorithm for thinning. [3]

1.2 Purpose

The main purpose and core objective of Pehchan is to facilitate various public and private sectors by changing a scanned Urdu document into an editable text file (which can be processed). Pehchan will be a good addition in the public sector but also its importance with other sectors couldn't be ignored. (Especially those sectors that are based on Urdu language automation)

1.3 Project Scope

All those requirements for the release 1.0 of Pehchan are covered in this document. Pehchan will allow the users to create an editable file for a scanned picture. The picture would have text written in Urdu with Noori Nastaliq font. Another advantage of Pehchan is that it will be helpful in the context that it will reduce the man power required to automate the Urdu records. It will provide the facility to edit the files those were previously saved as images. An alternative could be the typing of the whole Urdu record completely.

With reference to Pakistan and other countries where Urdu is used as the way of communication between public, its scope cannot be ignored.

Broadly speaking according to Pakistan Pehchan will be a good addition in the market related to public sector automation. Pakistan is a developing country and many of its

organizations (Government sector) are still manual, far from the emerging technologies in the field of information technology. These organizations have record from last 30-40 years and also before that on papers and this is the biggest hazard in the way of automation that how to automat by making all the record in papers from last half century computerized. In this regard Pehchan will prove itself as a good support for public sector automation.

1.4 Properties of Urdu Script

The modern Urdu alphabet consists of 39 basic characters. These characters are shown in Figure 1. In Urdu two or more characters may combine and create a complex shape character called compound character.

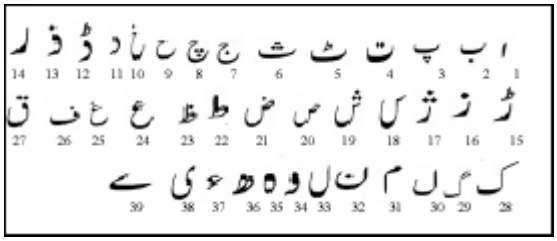

Figure 1 Urdu Alphabets

Some Also depending on the positions (first, middle or last) in a word the basic shape of a character may be changed. For example see Figure 2. Here an Urdu basic character in its isolated form and its shapes in first, middle and last positions of a word are shown. Writing style in Urdu is from right to left. [7]

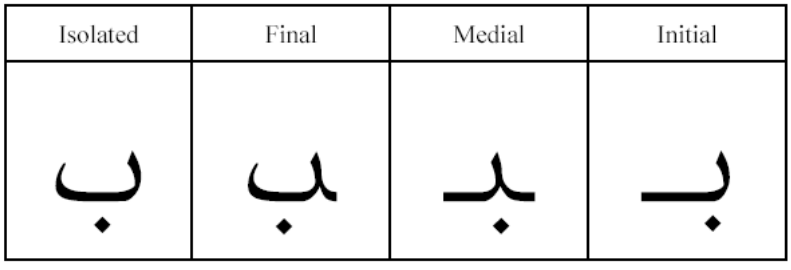

Figure 2 Shapes of an Urdu Alphabet

Chapter 2: Project Description

In this chapter we will discuss overall description of our project which includes product perspective, user classes and characteristics, operating environment, design and implementation constraints, user documentation and assumptions and dependencies.

2.1 Product Perspective

The product that we are going to establish is new and there is no such a product is in the market. This product is original and it is not a follow on member of any product family. This is not the replacement of any other software.

2.2 Product Functions

The complete detail of this section is discussed on the use cases sections.

2.3 User Classes and Characteristics

Two major Classes of users anticipated to use this product are

1. Frequent Users
    a. Public Sector Users
        i. Police Department Users
        ii. NADRA Staff
        iii. Judiciary Officials
        iv. The people containing the records of property (Patwari)
    b. Private Sector Users
        i. News Papers(Urdu)
        ii. Printing Press
2. Non Frequent Users
    a. Individuals
3. Project Team
    a. Project Supervisors
    b. Project Developers
    c. Project Testers

2.4 Operating Environment

This software is going to be used for converting Urdu language scanned image into an editable text file. Its platform is Microsoft Windows (Win XP, Win NT, Vista and latter). It coexists with majority of Microsoft applications.

2.5 Design and Implementation Constraints

General constraints include Economic, Political, Technical, System, Environmental, and Time & Cost. There is no political constraint involved in this project. But there is a time constraint, a short span of time (about 8 months). We also face some technical constraints regarding C# because it is new for us. English is be used for communication with the user as a language.

2.6 User Documentation

User Manuals will be provided with this software to work with in a much better manner. Tutorials will be provided if it would be possible with in this short span of time.

Chapter 3: Interfaces of Pehchan

External interface include user interfaces, hardware interfaces, software interfaces, assumptions and dependences, communication interfaces and specific requirements.

3.1 User Interfaces

Following are the external interfaces through which user can interact with Pehchan.

3.1.1 Splash Screen

"Every time a customer loads your application, you have the opportunity to impress or disappoint with your splash screen". [8]

Pehchan's splash screen appears when ever the Pehchan is loaded. It appears for few seconds and then disappears .Pehchan splash screen contains Pehchan logo and the names of team members.

Figure 3 Splash Screen

3.1.2 Main User Interface

The description of different parts of main user interface for Pehchan is as follow.

1. **File** menu consists of two options.
    i. Load image is used for loading the image from any location of the hard disk.
    ii. Exit is used for exiting the application.

2. **Preprocessing** menu consists of two options.
    i. Grey Scale Removal is used for removing the grey scale from image.
    ii. Noise Removal is used for removing the noise from image.
3. **Operations** menu consists of five options.
    i. Line Segmentation is used for segmenting the lines of text from image.
    ii. Ligature Segmentation is used for segmenting the ligature from line of text.
    iii. Ligature Thinning is used for the thinning of these ligatures.
    iv. Character Segmentation is used for segmenting characters from ligatures.
    v. Character recognition is used for recognizing these characters.
4. **Help** menu contains help about the use of Pehchan.
5. **About us** contains information about project team.
6. The Urdu image is loaded in this picture box for further processing. The image can be loaded by clicking the picture box.
7. This box shows the all the operations that have been performed.
8. The progress bar shows the progress of each operation during its execution

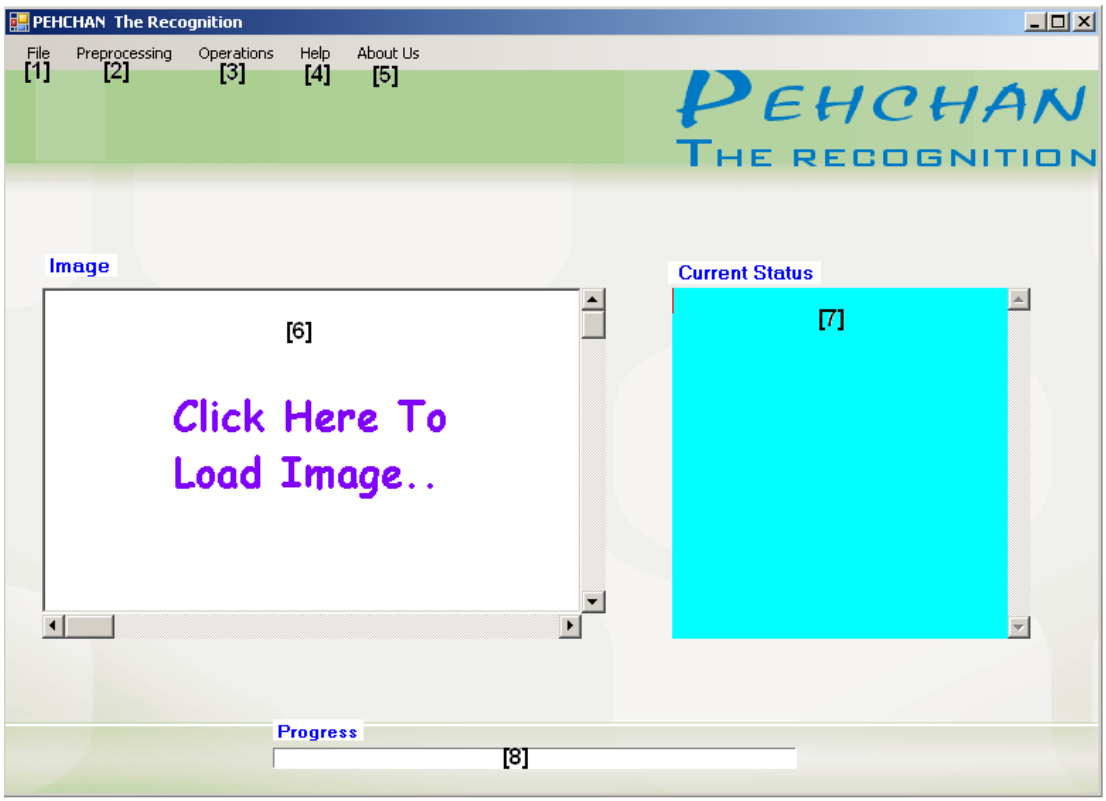

Figure 4 Main User Interface

3.2 Hardware Interfaces

Mainly there are two hardware interfaces. One is used for input and other is used for output. Beside these there are some other interfaces that are mentioned below. The system will make use of the operating system services to interact with the hardware (such as mouse, keyboard, monitor) installed on the user's machine. Apart the above ones we will also use printer and scanner in the system, as they are necessary. These hardware lines would be communicated through operating system.

3.3 Software Interfaces

Pehchan gets image from image repository. It can also get image from any specific location of hard disk.

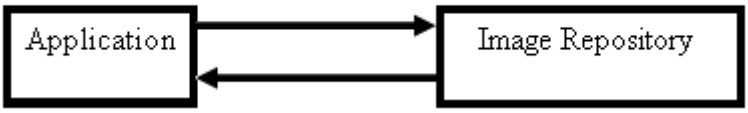

Figure 5 Software Interface

3.4 Assumptions and Dependencies

Following assumptions and dependencies are considered in Pehchan

1. Input image would be a monochrome image.
2. Input image would be a scanned picture of Urdu text.
3. Pehchan will run on Windows operating systems, Win XP and above.

3.5 Communications Interfaces

The front end of the software will be developed using the 8.0 version of .net studio environment. More over at the back end of the software there will be an image repository from where software can get images for further processing. Communication between software and image repository is done manually. User manually selects the folder of image repository for processing the images. Once folder is selected then Pehchan picks all images one by one for performing operations.

3.6 Road Map for Pehchan

In this section we will discuss specific requirements like image acquisition, image importing, image verification, matrix transformation, grey scale removal, dots removal and histogram formation etc. These requirements are describe with their inputs and outputs.

3.6.1 Image Acquisition

The first step is image acquisition. The input and output of this step are as follow:

Input

The input is picture containing Urdu text.

Output

The output is scanned file, which is stored on specific location on hard disk.

3.6.2 Image Importing

In this function Pehchan imports image for further processing. The input and output of this function are as follow:

Input

The input will be the scanned image that is stored on the hard disk.

Output

The output will be the image present in the image buffer for further processing.

3.6.3 Image verification

In this function Pehchan verifies whether image is present in the image buffer or not. Its input and output will be as follow:

Input

The input will be image present in the buffer.

Output

The output will be verified image buffer.

3.6.4 Grey scale removal

In this step Pehchan removes grey scale noise from the image. The input and output will be following:

Input

The input will be the digitized image with grey scale noise.

Output

The output will be image without grey scale noise.

3.6.5 Dots removal

In this function Pehchan removes unwanted dots from our image. The input and output will be the following.

Input

Input will be the image with dots.

Output

Output will be the image without unwanted dots.

3.6.6 Histogram formation

In this function Pehchan creates horizontal histogram of the image. The horizontal histogram is used for line segmentation. [5] The input and output of this function are as follow.

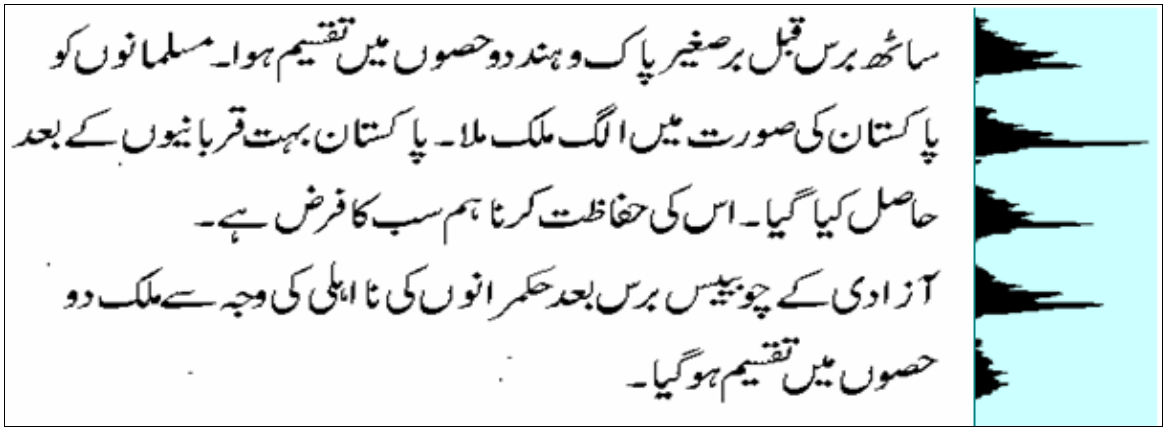

Figure 6 Horizontal Histogram

Input

Input will be image without noise.

Output

Image with corresponding histogram

3.6.7 Lines Segmentation

In this function Pehchan segments lines of text from input image. The input and output of this function are as follow:

Input

The input will be the image with corresponding histogram.

Output

The output will be the segmented lines of image.

3.6.8 Ligatures Segmentation

In this function Pehchan segments ligatures from each line of input text image. The input and output of this function are as follow:

Input

The input will be the segmented line of the image.

Output

The output will be the segmented ligatures.

3.6.9 Thinning

In this function Pehchan converts the segmented ligatures into their corresponding one pixel skeleton. The input and output of this function are as follow:

Input

The input will be the segmented ligatures.

Output

The output will be the thinned ligatures.

3.6.10 Character Recognition.

In this function Pehchan recognizes the characters using neural networks. The input and output of this function are as follow:

Input

The input will be thinned characters.

Output

The output will be recognized characters.

Chapter 4: System Features

System features include product functions, which are described with the help of use cases like image importing, matrix transformation, grey scale transformation and noise removal etc. Use cases are denoted by “UCph00”, where “UC” stands for Use Case and “Ph” stands for Pehchan (the name of project). In this notation two digits are used for numbering use cases.

UCPh01

4.1 Image Importing

4.1.1 Description

The use case gets an image from the specific location on the hard disk and then stores this image in the image repository.

4.1.2 Purpose

The purpose of this use case is to use the image for performing further operations on it.

4.1.3 Related use cases

The related use case in this case would be:

Figure 7 Use Case Diagram for image importing

4.1.4 Primary and secondary actor

No primary and secondary actor would be directly involved in this use case.

4.1.5 Pre condition

The pre condition in this use case is that the image should be present on the hard disk.

4.1.6 Main course

The main course of this use case will be as follow:

Select the required image from any location on the hard disk. After selecting the image place the image in image repository for further processing.

4.1.7 Alternative course

The alternative course would be followed if the image is not imported in the repository. In this case the appropriate error message will be generated that inform the user that image is not imported in the repository. Further processing can only be done if the image is present in the buffer.

4.1.8 Assumption

In importing image we will assume that the image contains the Urdu text.

4.1.9 Post condition

The post condition in this use case will be that the image would be present in the buffer for further processing.

UCPh02

4.2 Masking

4.2.1 Description

The use case will analyze a small part of the image at a time by selecting an appropriate mask size. The sample mask is shown below.

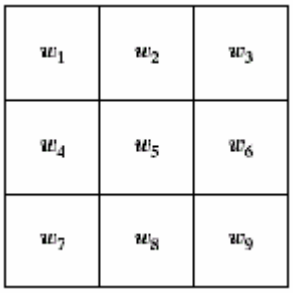

Figure 8 3X3 Mask

4.2.2 Purpose

The purpose of the use case is to help to analyze the image. After the use case we can check the image at regular intervals. Rather than being continuous the image will be digitized.

4.2.3 Related Use Cases

The use case will relate to the following use cases

1. Image importing
2. Gray level transformation

Figure 9 Use Case Diagram for Masking

4.2.4 Primary & Secondary Actors

No primary or secondary actors are directly related to this use case.

4.2.5 Pre Condition

The image would already be loaded into software.

4.2.6 Main course

The main course of the use case would be as follows.

1. Selection of the appropriate mask according to the text.
2. Deciding the operators of the mask.

4.2.7 Alternative Course

There is no alternative course of this use case.

4.2.8 Issues

No issues to be discussed in this use case.

4.2.9 Assumptions

No assumptions have been made regarding this use case.

4.2.10 Post Conditions

A matrix would be formed that could be used for the spatial filtering.

UCPh03

4.3 Gray Scale Removal

4.3.1 Description

This use case is basically for increasing the quality of the image by transforming that image into monochrome. There will be an initial value or threshold value against which each pixel of the image will be checked. If the value of the pixel in the image will be greater than or equal to the threshold value then value of that pixel will remain unchanged. If the value of the pixel will be less than the threshold value then value of that pixel will be set to white.

4.3.2 Purpose

The purpose of this use case is to facilitate the other use case like noise removal. So it is base for the noise removal. Noise removal will be efficient if the image is transformed into monochrome. It is pre processing for the use case of noise removal.

4.3.3 Related Use Case

The initiating use case:

The use case which will be initiated

4.3.4 Primary Actor

None

4.3.5 Secondary Actor

None

4.3.6 Precondition

The pre condition for this use case is that, image should be correctly captured and it should be free from errors.

4.3.7 Main Course

The main course of the use case is as follows:

1. Select a variable e.g. x to traverse image horizontally
2. Select a variable e.g. y to traverse image vertically
3. Select a Threshold value
4. Traverse first pixel from top left corner
5. Convert the color intensity of pixel between 0 and 1
6. Compare value of color intensity of pixel with Threshold value
7. If color intensity is greater then Threshold, convert the pixel to white color because it has tendency to white color
8. If color intensity is less then Threshold, convert the pixel to black color because it has tendency to black color
9. Repeat this procedure for all pixels of image

4.3.8 Post Condition

After the successful run of this use case, this use case should initiate to next use case which is for noise removal and for which this use case is designed.

4.3.9 Issues

Why we need to disturb the gray level of the pixel being scanned. We need to get the perfect image out the scrap so for that we have to define the approximately middle point i.e. threshold, so any point that is below it goes further below, and the one above it goes higher. Does it completely remove the point below the mentioned gray level? Yes it completely removes the point below the mentioned gray level.

4.3.10 Assumptions

1. The minimum size of the image is greater than 1 x 1
2. The image must have some distinct point, some distinct gray level unlike full white or full black
3. The Threshold value must be between the range of the minimum and maximum gray level of the image

UCPh04

4.4 Dots Removal

4.4.1 Description

This use case is basically for increasing the quality of the image by removing extra dots (those are not actual part of text) from image. These dots are actually noise in image. As this is noise therefore it produces problems in further processing e.g. a line with single dot

4.4.2 Purpose

The purpose of this use case is to facilitate the other use case like histogram formation. So it is base for the histogram formation. Histogram will be created efficiently if noise is removed from image. It is pre processing for the use case of histogram formation.

4.4.3 Related Use Case

The initiating use case:

Gray scale removal

The use case which will be initiated

Histogram formation

4.4.4 Primary Actor

None

4.4.5 Secondary Actor

None

4.4.6 Precondition

The pre condition for this use case is that, image should be transformed into monochrome.

4.4.7 Main Course

The main course of the use case is as follows:

1. Search the first row of pixels for black pixels
2. Once encountered
3. Check all neighbors of that pixel
4. If all neighbors are white then transform this pixel into white color
5. If one neighbor of the pixel is black then leave it as it is
6. Repeat procedure for whole image

4.4.8 Post Condition

After the successful run of this use case, this use case should initiate to next use case which is for histogram formation and for which this use case is designed.

4.4.9 Issues

Why it is need to remove dots from image? These dots are actually noise in image. This noise can produce problems in further processing. Therefore it is necessary to remove this noise.

UCPh05

4.5 Histogram Formation

4.5.1 Description

After the gray level image is converted in a monochrome image, this use case creates histogram so that line segmentation can be done.

4.5.2 Purpose

The purpose of this use case facilitates the use case of line segmentation. So this use case is base for line segmentation. If histogram is formed then line segmentation would be possible in Pehchan otherwise not.

4.5.3 Related Use Case

The initiating use case:

Gray Scale Removal

The use case which will be initiated

Line segmentation

4.5.4 Primary Actor

None

4.5.5 Secondary Actor

None

4.5.6 Precondition

The image should be optimized in terms of noise and variations.
The image should be greater than the size “1 x 1”.
The image should now be black and white

4.5.7 Main Course

The main course of the use case is as follows:

1. Initialize the counter with 0
2. Search first row of pixels of image for black pixels
3. Once encountered
4. Increment the counter
5. At the end of scanning of first row the counter has value equal to the number of black pixels in first row
6. Store value of counter in array of numbers
7. Repeat procedure for all rows of pixels of image

4.5.8 Post Condition

After the running of this use case there should be an array of numbers from which each number corresponding to the number of black pixels in each row.

4.5.9 Assumptions

The image is black and white.
The image has at least one line.

UCPh06

4.6 Line Segmentation

4.6.1 Description

This use case will divide the lines into different parts and make a separate image of each line from which we will separate each word and then characters from the words. So this is a linear operations and it is a chain of operations follow by one another.

4.6.2 Purpose

The purpose of this use case is to separate the lines in the image, because we can not separate a word from the whole image.

It is easy to handle lines for separating the words from lines so for this reason this use case has made.

4.6.3 Related Use Case

The initiating use case:

Histogram Formation

The use case which will be initiated

Ligature segmentation

4.6.4 Primary Actor

None

4.6.5 Secondary Actor

None

4.6.6 Precondition

The image should be optimized in terms of noise and variations
The image should be greater than the size “1 x 1”.
The image should now be black and white

4.6.7 Main Course

The main course of the use case is as follows:

1. Calculates the mean value of histogram
2. Initialize a counter with 0
3. Each row would be searched for number of black pixels
4. Start scanning the first row
5. Once encountered a black pixel
6. Increment the counter
7. At the end of scanning of first row compare value of counter with mean value of histogram
8. Continue this procedure for further rows of pixels
9. If value of counter become greater then mean value of histogram
10. mark it
11. Continue process of comparing
12. If value of counter become smaller then mean value of histogram
13. mark it
14. Now cut the image form first mark to this mark
15. Store it in array of images
16. Repeat the whole procedure for remaining rows of pixels

4.6.8 Post Condition

After the running of this use case there should be an array of images from which each image corresponding to the single line.

4.6.9 Issues

The only issue regarding the critical working of this use case is that there should be at least two black pixels between each full white row.

4.6.10 Assumptions

The image is black and white.
The image has at least one line
The image now has at least one black pixel bounded by full white rows

UCPh07

4.7 Ligature Segmentation

4.7.1 Description

This use case will take the image of line as an input and it will divide the line into further components, and each component will be an array. We will have an array of ligatures corresponding to a single line. Please note that the ligature means an isolated writing that may be an isolated character or any other menu script. This use case will lead us to next step which is character segmentation.

4.7.2 Purpose

The purpose of this use case is to transform the images of lines into array of images. Each line will be divided into ligatures. This use case is must for the character segmentation.

4.7.3 Related Use Case

The initiating use case:

Line segmentation

The use case which will be initiated

Character segmentation

4.7.4 Primary Actor

None

4.7.5 Secondary Actor

None

4.7.6 Precondition

The image should be optimized in terms of noise and variations.
The image should now be black and white.
The image being input to this use case is now image of a line

4.7.7 Main Course

The main course of the use case is as follows

1. For each image of the whole line repeat the following
2. Start scanning the image pixel by pixel
3. For each column from start to end
4. For each row start to end in the current column
5. If pixel is white skip it
6. If pixel is black then hold
7. Now once black pixel is encountered
8. If the pixel is alone then record this column as the starting column
9. Now go down pixel by pixel till another black pixel hit
10. If pixel is white skip it
11. If pixel is black then hold
12. If the pixel is not alone i.e. it has some neighbor to it except in the right direction
    a. Look for its neighbor on the below or on the left
    b. Follow the neighbor track of this single line pixel all directions except right
    c. When no more further pixel is found mark this column as the ending column
    d. Now cut the image between starting and ending column those are marked
    e. Put this image in the ligature image array
    f. And continue the whole algorithm for the rest of the image
    g. Now move to the next image in the lines image array till last
    h. Exit

4.7.8 Post Condition

After the running of this use case there should be an array of images from which each image corresponding to the ligature.

4.7.9 Issues

The use case is dealing with Marjory two types of the drawings i.e. a dot followed by a drawing or directly a drawing

4.7.10 Assumptions

The ligatures that start with such shapes that rotate to the write are not supported
The use case basically marks up the column number to be segmented

UCPh08

4.8 Thinning

4.8.1 Description

This use case is designed for the use case of recognition for which this use case is base. This use case will thin the pattern of ligatures found in the lines. This use case will make a single pixel of pattern found in the lines. This use case will be recursive as it will repeatedly work until no change is observed.

4.8.2 Purpose

The purpose of this use case is to thin up the pattern or ligatures and that is a must for the system as the detection use cases cant really work with the thick lines, as it is very difficult for them to study the features of the writing in a thick line as compared to the study of feature in thin line.

4.8.3 Related Use Case

The initiating use case

Ligature segmentation

The use case which will be initiated**:**

Feature Recognition

4.8.4 Primary Actor

None

4.8.5 Secondary Actor

None

4.8.6 Precondition

The most important pre condition for this use cases is perhaps the successful run of the Ligature segmentation use case. Because this use case can't work without any ligatures or patterns.

4.8.7 Main Course

The main course of the use case is as follows:

1. Define structuring elements as many as required they all are to be of the same size similar to the mask
2. Now start from the first structuring element till last,
3. Move the current structuring element from Start pixel till the last
4. If a match is found
5. Then close the pattern by a single pixel and move shift the structuring element one pixel ahead
6. Otherwise move the pixel ahead
7. Repeat the procedure till all structuring elements have been exhausted
8. Redo the whole use case till no change is observed

4.8.8 Post Condition

After the execution of this use case there should at least be no group of pixels found in the image which could further be thinned. At the end, ideally we should have a writing of single pixel.

4.8.9 Issues

How many structuring elements

Create two types of the structuring elements

One having the attributes of the object

The other having the attributes of the background

How to observe no change

When no structuring elements have a match then there is no further thinning possible

4.8.10 Assumptions

The image given to this use case is ligature segmented.

The image has some pattern unlike the full black or full white image

UCPh09

4.9 Set Input

4.9.1 Description

This use case is used for setting inputs for input neurons. Some input is assigned to input neurons before initiating of neural network training and learning process. Each input neuron is given a code based on feature set.

4.9.2 Purpose

The main purpose of this use case is to give inputs to the input layer of network so that it starts its working.

4.9.3 Related Use Case

Feature extraction use case and image acquisition use cases are the related with this use cases.

4.9.4 Primary & Secondary Actors

There are no primary and secondary actors related to this use case.

4.9.5 Pre condition

Features of Character must be extracted so that they assign some specific code and ready for recognition.

4.9.6 Main Course

The main course of this use case is as follows

1. Specific inputs are assigned to the input neurons
2. Input is in the form of numeric codes
3. Each input neuron has different value from other

4. When an extracted feature is come on input layer its code is matched with input neuron value if it is matched with any of input neuron value then this extracted feature has this line in network.

4.9.7 Input Parameter

There are no input parameters to this use case by any actor

4.9.8 Output Parameter

There are no output parameters to this use case by any actor

4.9.9 Post Condition

After inputs are provided on the input neurons Load Net (Activate) use case is started.

4.9.10 Alternative Course

There will be no alternative use case for this use case.

4.9.11 Issues

There will be question why you provide numeric input to input layer of neural network? Because neural network can only accept inputs on input neuron in numeric form no other input method is taken as input on this layer.

4.9.12 Assumptions

There will be no assumption related this use case.

UCPh10

4.10 Load Net (Activate)

4.10.1 Description

This use case is used for loading the network or initiating the network to start its working Network will start training and learning after this use case.

4.10.2 Purpose

The main purpose of this use case is to initiate the process of network training and learning

4.10.3 Related Use Case

This use case relates with all other use cases because function of this use case is to initiate the network.

4.10.4 Primary & Secondary Actors

There are no primary and secondary actors related to this use case.

4.10.5 Pre condition

Inputs must be assigned to input neurons so that network is loaded with correct values.

4.10.6 Main Course

The main course of this use case is as follows

1. Network is initiated so that it starts its works and start performing calculation for middle layer and out put layer
2. After loading network starts its working and remain in this mode until output is matched or exit is called

4.10.7 Input Parameter

There are no input parameters to this use case by any actor

4.10.8 Output Parameter

There are no output parameters to this use case by any actor

4.10.9 Post Condition

After this use case network starts its working.

4.10.10 Alternative Course

There will be no alternative use case for this use case.

4.10.11 Issues

There will be question why you load your network?
Because network can not starts its working process prior to loading.

4.10.12 Assumptions

Their will be no assumption related this use case.

UCPh11

4.11 Calculating Middle Layer Input

4.11.1 Description

The role of this use case is to calculate the middle layer values .Input neurons values are multiplied with their respective weights and then the resultant is given as input to hidden layer.

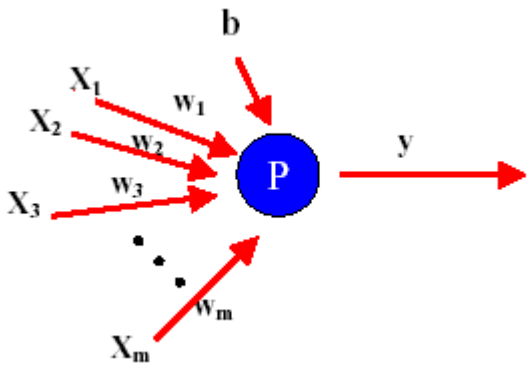

Figure 10 Middle Layer Inputs

4.11.2 Purpose

The purpose of this use case is to calculate the middle layer values which are used by further layers.

4.11.3 Related Use Case

This use case relates with calculate output for output layer.

4.11.4 Primary & Secondary Actors

There are no primary and secondary actors related to this use case.

4.11.5 Pre condition

Inputs must be assigned to input neurons so that network is loaded with correct values.

4.11.6 Main Course

The main course of this use case is as follows

1. The main purpose of this use case is to calculate the values for middle layer input
2. Input neurons are multiplied with their weights

3. Resultant values are applied to the middle layer which is input for transfer function.
4. These weights are updated when results of out put layer are not correctly identified.

4.11.7 Input Parameter

There are no input parameters to this use case by any actor

4.11.8 Output Parameter

There are no out put parameters by this use case which is shown to the actor

4.11.9 Post Condition

After this use case network starts its working.

4.11.10 Alternative Course

There will be no alternative use case for this use case.

4.11.11 Issues

There will be a question why middle layer values are calculated?
Middle layer values are calculated for the input of hidden layers input neurons are multiplied with their respective weights which are input for hidden layer.

4.11.12 Assumptions

Initial weights are assigned to all layers .When network is loaded input neurons are multiplied with these weights.

UCPh12

4.12 Calculation of output layer input

4.12.1 Description and Priority

It is an essential step in neural networks training. It will format the output of middle layer for output layer. By formatting, I mean weighted sum of inputs (with bias), which is required for output layer. Every layer has an input domain and that layer doesn't support the input which is not in that range (i.e. between 0 and 1). The following diagram will give you a brief view of this system feature.

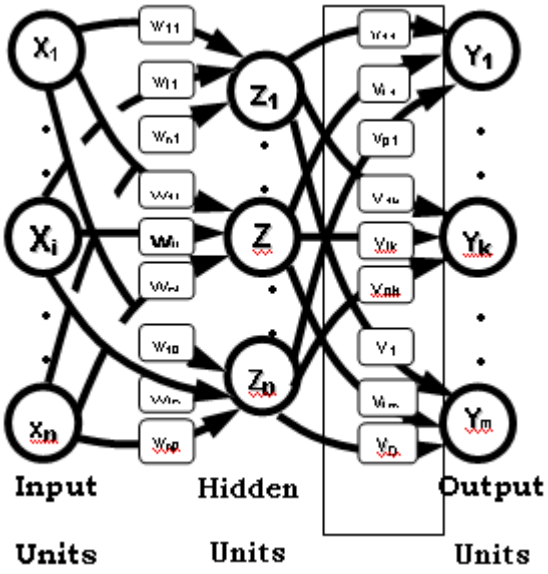

Figure 11 Calculation of output layer input

The image above describes the feature, the output of the hidden units is multiplied by the specified weights (i.e. v1….vn), which makes the input of the output layer. Weights are adjusted according to the output layer but this is the issue of weight calculation system feature.

4.12.2 Purpose

Purpose of this feature is to calculate the input for output layer or the output of hidden layer, so that the input is formatted for output.

4.12.3 Related Features

Its feature will include other features (use cases) like weight calculation, weight specification after each iteration or training step.

4.12.4 Primary Actor

None

4.12.5 Secondary Actor

None

4.12.6 Pre-condition

Working Condition:

The system must be in working condition and initialized to start the application.

Weights Specification:

Weights must be specified before this feature, because it needs weights for calculations. So the weights must also be updated after each iteration.

Hidden Layers:

Hidden layers have to work properly to execute this feature. Work properly means that the values calculated by hidden layer through activation function must be correct (i.e. it should be in the range/domain).

4.12.7 Stimulus/Response Sequences

User Actions:

None.

System responses:

System will not response to user after the execution of this feature, it will give the results to output layer of neural net.

4.12.8 Main Course

The main course of this feature is as follows.

1. Take as input, the output of hidden units.
2. Multiply the weights specified at each edge with inputs from hidden units.
3. Calculate the input for the output layer (i.e. weighted sum of all inputs at each unit).
4. Verification of values calculated for the input of output layer.

4.12.9 Post-Condition

After the values are calculated and input is formatted for the output layer. These inputs are given to the output layer, which will then generate final output (of each iteration).

4.12.10 Alternate Course of Event

This system feature will execute once for each iteration, if the values calculated are not valid, it will not execute again or no other system feature will execute in that case. So no specified alternative course of event will be followed in this case.

4.12.11 Issues

What will be the case if, values calculated by this feature doesn't lie in the range of output layer units?

In this case this system feature will not solve the problem. It will only calculate the values and from that value it can be verified that the values are in the range of output units or not. If values are not in the specified range then there must be a problem in weight specifications, so this problem will be solved by weight specification feature.

4.12.12 Assumptions

Following assumptions are made in execution of this feature.

1. Weights which are specified are correct and they let the value be in the range of output units.
2. Values calculated by hidden layer or activation function are also correct.

UCPh13

4.13 Activation Function Calculations

4.13.1 Description and Priority

It is an essential step in neural networks training. It will calculate output of middle layer. The function used to calculate the values are activation functions. A function used to transform the activation level of a unit (neuron) into an output signal. Neural Networks supports a wide range of activation functions, i.e. Identity functions, Logistic functions, Hyperbolic functions, Exponential functions, Softmax functions etc. It will take as input the values calculated after input layer (PSP). It will produce the output which is use by "Calculation of middle layer output" feature to calculate the input for output layer.

The following diagram will give you a brief view of this system feature.

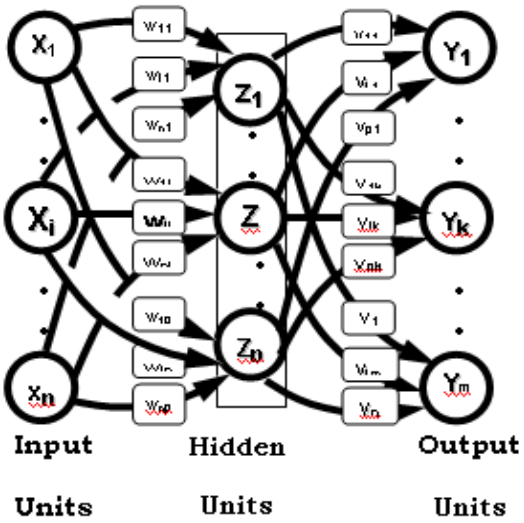

Figure 12 Activation Function Calculations

The image above describes the feature. The activation function is calculated at hidden units (units in hidden layer), the area enclosed by a rectangle.

4.13.2 Purpose

Purpose of this feature is to calculate the output of the hidden layer which is used in calculating the input for output units (units in output layer).

4.13.3 Related Features

This feature doesn't include other features.

4.13.4 Primary Actor

None

4.13.5 Secondary Actor

None

4.13.6 Pre-condition

Working Condition:

The system must be in working condition and initialized to start the application.

PSP calculation:

PSP (Post-Syanaptic-Potentional) calculation must be done before the execution of this feature. PSP will format the input of activation function.

Activation function definition:

Activation function must be designed to produce valid output, i.e. the output must be in the range. So that after calculation it is acceptable for the output layer as input.

4.13.7 Stimulus/Response Sequences

User Actions

None.

System responses

System will not response to user after the execution of this feature, it will give the results to "Calculation of middle layer output", which give input to output layer.

4.13.8 Main Course

The main course of this feature is as follows.

1. Take as input, PSP (weighted sum of inputs - threshold).
2. Verify the input that it is the range of activation function or not.
3. Calculate the value of activation function, applying the PSP value on it.
4. Give the output to "Calculation of middle layer output" feature (after verifying the range).

4.13.9 Post-Condition

After the activation function values are calculated and input is formatted for the calculation of values for (input of) output layer. These inputs are given to the output layer, which will then generate final output (after each iteration).

4.13.10 Alternate Course of Event

This system feature will execute once for each iteration, if the values calculated are not valid, it will not execute again or no other system feature will execute in that case. So no specified alternative course of event will be followed in this case.

4.13.11 Issues

What will be the case if, values calculated by this feature doesn't lie in the range of output layer units?

In this case this system feature will not solve the problem. It will only calculate the values and from that value it can be verified that the values are in the range of output units or not. If values are not in the specified range then there must be a problem in activation function specifications or weights calculation before the activation function execution, so this problem can be solved by reconsidering the activation function and the weights calculation methodology.

4.13.12 Assumptions

Following assumptions are made in execution of this feature.

1. Activation function that is specified is correct, i.e. it will produce valid output (acceptable for output layer calculations).
2. PSP calculated for this feature is also correct.

UCPh14

4.14 Training

4.14.1 Description and Priority

It is a most essential feature of neural networks. It is the overall working of the neural network. It can be better explained by a number of steps as follows

1. Input to the input layer units (features extracted from the character segments).
2. Calculations on input units to produce the inputs for hidden layer (i.e. PSP calculation).
3. Calculations of activation function values to produce output of hidden layer.
4. Calculations on hidden layer output to produce input for output layer.
5. Final output from output layer.
6. Calculation of error function.
7. Adjustment of weights according to the error.
8. Repeating the whole process again.

The following diagram will give you a brief view of this system feature.

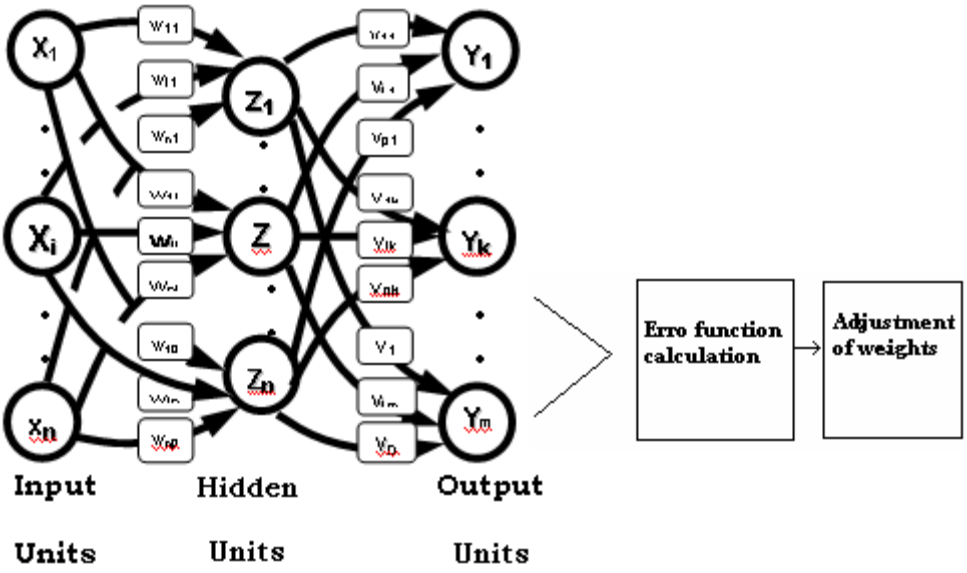

Figure 13 A Simple Neural Model With One Hidden Layer

The image above describes the feature. Every feature related to it is explained as a separate feature in this document.

4.14.2 Purpose

Purpose of this feature is to train the inputs on the network. Input set contain many cases (every possible case in the product domain) and network will be trained for every case (input).

4.14.3 Related Features

This feature include many other features

1. Load Net
2. Calculating Output of Input Layer
3. Calculating Activation Function
4. Calculating Input of Output Layer
5. Error Calculation
6. Adjust Weights
7. Save Net
8. Stop Working

4.14.4 Primary Actor

None

4.14.5 Secondary Actor

None

4.14.6 Pre-condition

Working Condition:

The system must be in working condition and initialized to start the application

4.14.7 Feature numbers calculation:

Unique feature numbers according to the range of input units must be generated before execution of this feature. We can also say this step as verification of inputs to the input units.

4.14.8 Initial Weights Specification:

Before executing this feature the initial weights must be specified. After each iteration weights will be adjusted but initial weights must be specified for start of training.

4.14.9 Activation function definition:

Activation function must be designed to produce valid output, i.e. the output must be in the range. So that after calculation it is acceptable for the output layer as input.

4.14.10 Threshold and Bias values specification:

A thresh hold and Bias values must be specified for the hidden units (hidden layer).

4.14.11 Error Function specification:

Error function must be specified before execution of this feature, as it calculates error after each iteration to adjust the weights. Error function specification also depends on transfer function specified.

4.14.12 Stimulus/Response Sequences

User Actions:

None

System responses:

System will not respond to user after the execution of this feature, it will train the network on different inputs, so nothing to do with user.

4.14.13 Main Course

The main course of this feature is as follows.

1. Take as input the feature numbers which are generated according to the features extracted from the character (segmented) and give it to the input layer units.

2. Perform calculations on input units to produce the inputs for hidden layer (i.e. PSP calculation).
3. Calculate activation function values to produce output of hidden layer.
4. Perform calculations on hidden layer output to produce input for output layer.
5. Apply error function on outputs of output layer and calculate error.
6. Adjust the weights according to the error produced.
7. Repeat the whole process with new weights (if error produced).
8. Stop the process when error becomes zero (negligible).

4.14.14 Post-Condition

After the execution of this feature on every input (in the set), the neural net will ready to go for learning on new inputs. After training the product can be executed with its full potential.

4.14.15 Alternate Course of Event

This system feature will be executed once for each input. It is essential for each input. So no specified alternative course of event will be followed in this case.

4.14.16 Issues

What will be the case if, neural network takes too long to train inputs or some specific input?

In this case this system feature will not directly solve the problem. But it will work with improved learning rate. Error function after each iteration calculates the error, which determines the error surface, so by observing the error surface learning rate of the network can be improved.

4.14.17 Assumptions

Following assumptions are made in execution of this feature.

1. Feature numbers generated are valid for the input layer and unique.
2. Output classes (desired outputs for each class) are specified in the range which can be produced by the output layer.

UCPh15

4.15 Error Calculation

4.15.1 Description

The use case is used to calculate the error in case of back propagation. It is the use case to initiate the learning process of the MLP.

4.15.2 Purpose

The purpose of the use case is to reduce the chances of miss classification.

4.15.3 Related Use Cases

The use cases related to this use case are

1. Calculate output.
2. Adjust weights.

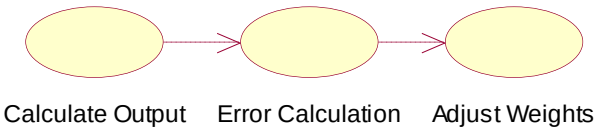

Figure 14 Use Case Diagram for Error Calculation

4.15.4 Primary & Secondary Actors

No primary and secondary actors involved in this use case.

4.15.5 Pre Condition

The pre condition is that the output has been calculated before initiating this use case.

4.15.6 Main Course

The main course of the use case is as follows.

1. Find out the current weights.
2. Find the difference between the class value and the deviation value.
3. Change the weights accordingly.
4. Increment the iteration number.

4.15.7 Alternative Course

No alternative course of the use case.

4.15.8 Issues

No issues relating the use case.

4.15.9 Assumptions

No particular assumptions.

4.15.10 Post Condition

The post condition is that the miss classification error has been found.

UCPh16

4.16 Weights Adjustment

4.16.1 Description

The use case is used for the learning process. In this use case we tend to find out the weight adjustment quantity.

4.16.2 Purpose

The purpose of the use case is to reduce the chances of miss classification next time. This time it has learned a new shape by adjusting the weights. Next time this shape will not be a new one for the use case.

4.16.3 Related Use Cases

The use cases related to this use case are

1. Error Calculation
2. Save Net

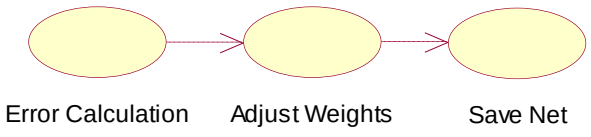

Figure 15 Weights Adjustment Use case

4.16.4 Primary & Secondary Actors

No primary and secondary actors involved in this use case.

4.16.5 Pre Condition

The pre condition is that the output has been found as the execution results of the last iteration.

4.16.6 Main Course

The main course of the use case is as follows.

1. Find out the current weights.
2. Find the difference between the recommended & current rates.
3. Change the weights accordingly.

4.16.7 Alternative Course

No alternative course of the use case.

4.16.8 Issues

No issues relating the use case.

4.16.9 Assumptions

No particular assumptions.

4.16.10 Post Condition

The post condition is that the new weights have been found.

UCPh17

4.17 Save Net

4.17.1 Description

The use case is used to save the learning process that has been adjusted in the learning process in the form of weights adjustment.

4.17.2 Purpose

The purpose of the use case is to keep the weights in the memory that has been adjusted.

4.17.3 Related Use Cases

The use cases related to this use case are

1. Adjust Weights

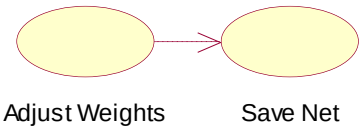

Figure 16 Use Case for Save Net

4.17.4 Primary & Secondary Actors

No primary and secondary actors involved in this use case.

4.17.5 Pre Condition

The pre condition is that the new weights have been calculated.

4.17.6 Main Course

The main course of the use case is as follows.

1. Check that weights have been adjusted or not.
2. If yes then store the new ones.

4.17.7 Alternative Course

No alternative course of the use case.

4.17.8 Issues

No issues relating the use case.

4.17.9 Assumptions

No particular assumptions.

4.17.10 Post Condition

The post condition is that the new weights have been saved on the net.

UCPh18

4.18 Stop working

4.18.1 Description

This use case basically helps in keeping the neural network up to date. The weights of initial, middle, and final layer. All these weights are updates every time the neural network goes through the whole rotation. Through this use case the processing out put use case is also initiated which is given the internal output of the neural network which is mapped accordingly to the Unicode. This use case also stops the neural network just to initiate the processing output, and then resumes the neural network again for next detection. The neural network may not require re initialization etc stuff to move on.

4.18.2 Purpose

The basic purpose of this use case is to preserve the updated working stage of the network so that the network does not go through the rigorous training again and again

4.18.3 Related Use Case

The initiating use case:

Error calculation

The use case which will be initiated:

Processing output

4.18.4 Primary Actor

None

4.18.5 Secondary Actor

None

4.18.6 Precondition

The precondition includes

1. The neural network is initialized once.
2. The neural network is in working and trained condition.

4.18.7 Main Course

The main course of the use case is as follows

1. The error correction gives the internal code of the neural network as argument to this use case
2. The weights, updated ones are stored so that the neural network gets faster
3. The process output is initiated with the internal code as argument
4. Resume the neural network for normal processing
5. Exit

4.18.8 Post Condition

The post conditions to this use case are as below

1. The network will be in working condition again
2. The processing output will have successfully get the argument in form of internal code

4.18.9 Issues

The major issues regarding this use case are as below

1. The neural network gives it output to the character wise
2. Each Unicode is given as argument to the processing output
3. The neural network should resume accordingly

4.18.10 Assumptions

The only assumptions to the both sides of the road is that

1. Neural network is giving respective output and processing output getting the respective inputs
2. Between the passing of the arguments the neural network goes in the ideal state

Chapter 5: Non Functional Attributes of Pehchan

Other nonfunctional features include performance requirements, safety requirements, security requirements, software quality attributes, Packaging Requirements and Business Rules

5.1 Performance

The system responds to each user input within specific time. The system is able to handle scanned Urdu written text image with noise, or noise which will be produced during scanning the image. The system removes the noise from that image. In this way the quality of image will be increased. Time is the main factor involved in performance. The system fulfills every desired task in the minimum time. Application is designed for the single user.

5.2 Safety

When the application does fail, the user can expect no harm done to the application itself, only to the data objects that had been modified but not saved. The application will not have a single point of failure.

5.3 Security

Users will only access data and services for which they have been properly authorized.

5.4 Software Quality Attributes

Pehchan fulfills software quality attributes like reliability, availability, security, maintainability, and portability.

5.4.1 Reliability

Pehchan will be reliable during all the stages. Working on one stage after another will be in an efficient way. There will be no chance of loosing data at any stage due to working of application.

The system will never crash or hang, other than as the result of an operating system error. The system will provide graceful degradation in the face of network delays and failures, or when handling large size noisy images.

5.4.2 Availability

There are no specific availability requirements for Pehchan. Pehchan will be available for its user at anytime. Once the image is scanned, it will be stored in hard disk of the computer and it can be utilized at any time. And no need to scan it again. When user wants to use the application, then he will get image from hard disk at anytime.

5.4.3 Security

User can only scan and edit the image but user will not be allowed to edit the application and make any changes in the code of the Application. User will only have an executable form of the application, where the code will not be available to users of the software for making any change; also users can not view the code. Only the developers of the software will be allowed to enhance or make a change in the functionality of the software according to need of the users.

5.4.4 Maintainability

All code will be fully documented. Each function will be commented with pre- and post-conditions. All program files will include comments concerning authorship and date of last change. So in the future it will be easy to edit or add new functionalities in the application.

The code will be modular to permit future modifications. Pehchan is based on object oriented features so any new changes in its functionality can be added with the need and time.

5.4.5 Portability

Pehchan is designed for the Microsoft Windows (Windows implementations shall be portable to all versions of Windows up to and including Windows XP). Application will be easy to install and will be portable to any Microsoft Windows operating System.

5.5 Packaging Requirements

The system will be packaged along with source code and all documentation, and will be available for electronic transfer as a single compressed file. The uncompressed set of files will include a README file containing a minimal guidance for installing and running the software, including recompilation if needed. If recompilation is necessary during installation, the system shall include a make file

.

Appendix A: Glossary

Editable A files which can be changed or processed and allow formatting options.

NADRA National Database Registration Authority

Noori Nastaliq is cursive Urdu font

OCR Optical Character Recognition

Pehchan is an Urdu word which means to recognize. It is the name of our software.

References

[1]. B B Chaudhuri, U Pal And M Mitra. Automatic recognition of printed Oriya script. Computer Vision and Pattern Recognition Unit, Indian statistical Institute, 203, B T Road, Kolkata 700 108, India

[2]. Digitization Center. The University of Texas at Austin. April 2004

[3]. Dr. John Cowell , Dr. Fiaz Hussain. Thinning Arabic Characters for Feature Extraction

[4]. Liana M Lorigo. Offline Arabic Handwriting recognition:A survey. Transactions on patteren analysis and machine intelligence in IEEE.

[5]. M.Salmani Jeloder, M. J. Fadaeieslam, N. Mozayani , M.Fazeli. A Persian OCR system using Morphological operators. Transactions on engineering , computing and technology V4 Feb 2005.

[6]. Muhammad S. Khorsheed , William F Clocksin. Structural Features of Cursive Arabic Script

[7]. S. Chandana, U. Pal. English, Devangari and Urdu Text Identification. Proceedings of the international conference on cognition and recognition.

[8]. Tom Clement. A Pretty Good Splash Screen in C#. Available from www.codeproject.com. Accessed 10 August 2007.

[9]. Tom Mochal. Think Ahead: Defining Your Testing Strategy. Available from http://articles.techrepublic.com.com/5100-22-1058735.html (2001); Accessed 10 August 2007.

[10]. U.Pal, Anirban Sarkar. Recognition of Printed Urdu Script. Proceedings of the 17th international conference on document analysis and Recognition 2003.