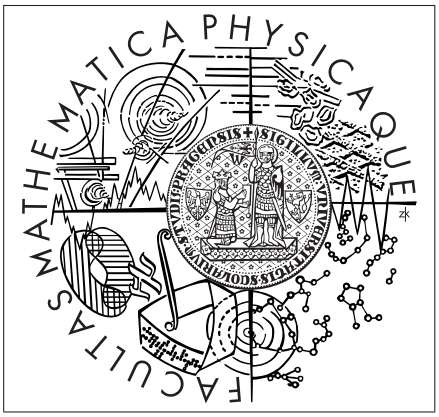
Charles University in Prague
Faculty of Mathematics and Physics

DIPLOMA THESIS



Prajol Shrestha

Online and Offline Vocabulary Correction while using Digital Pen

Institute of Formal and Applied Linguistics

Supervisor: Prof. Abdel Belaïd (University of Nancy 2)
Co-Supervisor: Dr. Václav Hlaváč (Charles University)
Study Program: Computer Science
Study Specialization: Mathematical Linguistics

European Masters Program in Language and Communication Technologies (LCT)

2009

Contents

Declaration

I hereby declare that this diploma thesis is my own work and where it draws on the work of others it is properly cited in the text. I understand that my thesis may be made available to the public.

Prague 04/08/2004 **Prajol Shrestha**

Abstract

Title: Online and Offline Vocabulary Correction while using Digital Pen
Author: Prajol Shrestha
Department: Mathematics and Physics
Supervisor: Prof. Abdel Belaïd
Co-Supervisor: Dr. Václav Hlaváč
Supervisor's e-mail: Abdel.Belaid@loria.fr
Co-Supervisor's e-mail: hlavac@fel.cvut.cz

This thesis is a requirement for the completion of my double degree with Charles University and University of Nancy (France) under the Erasmus Mundus masters program in Language and Communication Technology. The topic was provided by Prof. Abdel Belaïd along with Actimage Company and has been completed and successfully defended at University of Nancy under their supervision. After the defense in France, I came here to Prague and worked with my co-supervisor Dr. Václav Hlaváč to improve the text of the thesis to meet the standards of Charles University in order to defend it in Prague.

The aim of this work is to study some helpful handwriting correction marks and propose a system that automatically incorporates the corrections made by the writer while writing with a digital pen. The problem is complex because the corrections have to be at the same time readable by the writer and by the machine. Even detecting a free flowing line of handwritten text has not been understood completely by machines. The spatial relations between the correction marks and the text line are easily detected by the writers and readers and hence systematic to them, yet for machines this detection is a huge challenge. Humans have an intelligence that is hard to mimic by machines. This seemingly systematic writing has not been studied in abundance. Even though the literature is abundant of printed document annotation, there are few researches which are directed towards the correction of digital text using handwritten correction marks but none is present for free flowing handwritten text. This study is proposed by the Actimage Company aiming to offer to her customers a helpful tool

for automatically converting handwritten text with corrections to digital data in the form of image and digital text using their product Actinote©.

Chapter 1

Introduction

In this digital age, we have been highly dependent on digital devices as it has made life easier. We use digital devices in abundance starting from computers to mobiles because of efficient and practical way in which it helps us complete our task.

Among the digital devices, one of the product which is emerging as a popular device is the digital pen [4]. Digital pen is a traditional ball point pen which writes on paper and captures various types of information as the user writes. This information can be used to get digital copies of what has been written in the form of an image and possibly as text. These pens are used in several applications including taking notes, filling in forms and even sending emails. We could imagine many situations where these pens could be used, such as in a meeting where it aids people to communicate easily, or it could allow an employer or an engineer to register notes about a product, or even in outdoors where an architect uses it to sketch the plan of a building. The information in the pen is then transferred to a storage device like a computer or a PDA via Bluetooth technology or through an USB connection. This information is then processed and provides the user with the corresponding recognized text and an exact image of what was written.

The use of the digital pen in daily life introduces situations where mistakes occur while writing. In such situations, the writer makes corrections to correct those mistakes during the writing process or after finishing the writing to indicate what was actually intended. Corrections such as deletion or movement could be made on characters or words. In the present scenario, an exact image of what was written is generated without the corrections made and the recognition system is unable to recognize the text properly due to the non character strokes. A stroke is the information written between the time of the pen down till pen up. It would be desirable and practical to receive an image and a recognized text which resembles what the user intended to write. This work deals with this aim of

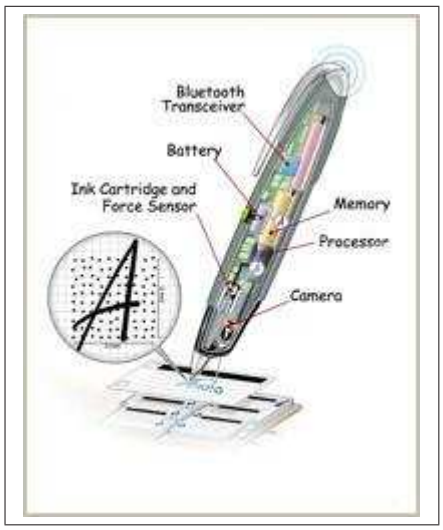


Figure 1.1: The Anoto pen, http://www.anoto.com/the-pen.aspx

incorporating the corrections on to the image and the recognized text.

Actimage, with whom we are collaborating, offers her clients new interactive human machine solutions based on the Anoto digital pen [1]. The digital pen is equipped with ink, a small digital camera, memory to store up to 40 pages and is about the size of a highlighter pen [8] (see Figure 1.1, which illustrates the Anoto pen). The digital pen writes on a special paper termed as the digital paper [3], also known as the interactive paper, which has all the properties of a normal paper with an addition of a series of tiny dots pre-printed on to it. These dots, which are spaced about 0.3 mm apart, have a pattern which uniquely identifies the position coordinates on the paper. The digital pen uses this pattern to store the x and y coordinates of the handwritten strokes and uploads it to a computer. Throughout the writing, the small digital camera integrated into the pen continuously takes pictures of the paper and its grounds for the digitization of the written information.

Actimage provides a system for taking notes, filling in forms, etc. This system stores the written information and generates an image with the possibility to automatically recognize the writing by an OCR. The image can be sent to the server via Bluetooth which allows the user to automatically check his document in an online or offline manner.

1.1 Objective

Nowadays, the system that Actimage provides is able to recognize the handwriting in the image generated by the pen but there is no mechanism to handle corrections. The image, with the corrections along with the handwritten text,

as the input for the OCR is different from its usual input of only handwritten text which prevents the OCR to perform at its best. This is why the company is seeking an automatic correction system to remedy this drawback. The objective of this work is to propose a system by proof of concept in a short period of time of four months that will automatically recognize the corrections made by the user and incorporate them into the image generated by the digital pen. This incorporation of the corrections will generate a more readable and presentable image of the written information for both humans and machines. This image can then be sent to an OCR for generating the digitized text of the image.

This system that we propose has to be able to work in both the online and off-line mode. The on-line and off-line in the context of digital pen differs from its traditional sense in the area of handwriting recognition systems. On-line handwriting recognition system recognizes text as the user writes. The system captures writing process using a digitizer such as special pens and pads touch sensitive screens or vision-based pen tip trackers. They encode the dynamic information of the writing process in addition to the static physical layout and the data is stored as a sequence of strokes. Whereas off-line handwriting recognition systems recognizes the text after it has been written. This system uses digitized images using scanners or digital cameras. The information stored in the off-line system is usually the pixel information of the image [14] [27] [13]. With these definitions our digital pen would be an on-line system.

The on-line and off-line concept in our system indicates the time of correction. The correction done during the writing and before sending the information to the computer is the on-line mode while the off-line mode is the correction done on a document whose information has already been sent to the computer (see in Figure 1.2).

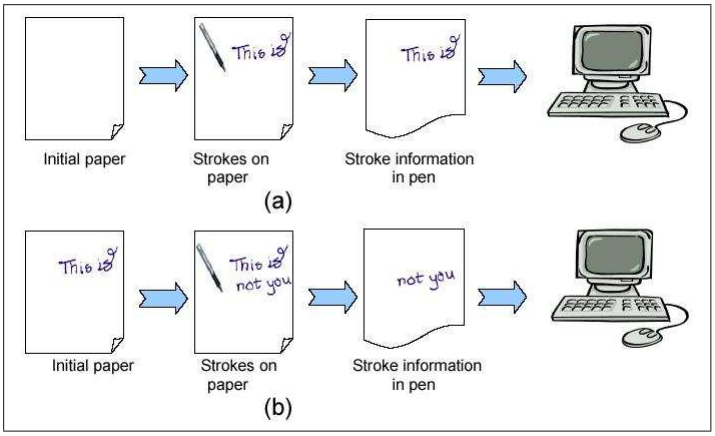


Figure 1.2: Information transfered by the pen in the (a) Online and (b) Offline mode.

System for editing free flowing handwritten text does not exist, but there

already exists some editing systems that edit digital documents such as typed text documents [23] [16] [24] [13] and web pages [38]. Most of these systems have a purpose of removing the use of paper in editing processes, but some offer paper editing techniques as well. These existing systems that only deal with editing digital document can easily detect the correction symbols from the text because their x and y coordinate information are naturally separated. This information is present in different files or in different colors which are different in our case, where the information of the corrections and the handwritten text is in the same file with no additional information to distinguish them.

There are two major parts in building this system. One is the standard to use a standard set of correction symbols and the other part is the recognition of those symbols.

The set of correction symbols are defined for vocabulary corrections and are selected from existing correction guidelines [20] with the introduction of some new symbols. Each correction symbol has its own defined shape and spatial context which are used for recognition and incorporation respectively. The recognition system uses fuzzy sets which correspond to unique features extracted from the correction symbols [43]. These features have been studied carefully in order to be representative of each symbol. Each symbol is present in every fuzzy set with some membership value assigned to it by the fuzzy functions which are also known as the membership functions. These fuzzy sets are used for the fuzzy classification process as a proof of concept. We have chosen 3 simple methods to implement the classification and compared the results to propose the best method. These algorithms are decision tree, min-max, and fuzzy neural network.

This report is organized as follows: we start by presenting the state of the art in chapter 2. In chapter 3, the possible correction scenarios and the correction symbol are discussed. In Section 4, the correction process is discussed in detail. Section 5 shows the experiment and its results and finally in section 6 we conclude by giving our perspective on this system.

Chapter 2

Standards for Corrections

Corrections made on paper can be of two types: vocabulary corrections or structural corrections. Vocabulary corrections are corrections that correct the spelling of words (see Figure 2.2) whereas structural corrections deal with the structure of the written text i.e. indentations, movement of paragraph. Proof reading handles these two types of corrections. There are many guidelines for proof reading correction symbols [20]. The basic idea for the standard proof correction marks is for its easy understanding of the mistakes made and its corresponding corrections by everyone who does manual text editing. We focus our correction system on correcting vocabularies.

The corrections are made up of two possible items. One is the correction symbols/marks that indicate the mistakes, the context of the symbol, and the action to be taken for its correction and the other item being the corrections for that mistake (see Figure 2.1).

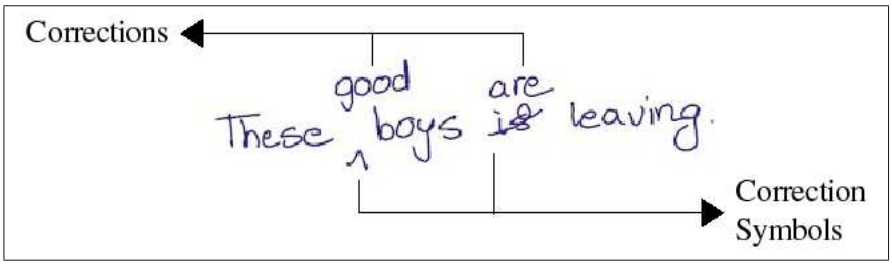


Figure 2.1: Correction symbols and corresponding corrections

There are many correction symbols to indicate what mistakes were made for example, the caret sign to indicate the missing character or word. These correction symbols are also called proofreading symbols as it is used by proof readers for proofreading [20] [18]. There are also standards on how corrections should

be indicated. Different organizations have their own proofreading standards, for instance the Czech norm, ČSN 880410 [22], as well as the German norm, DIN 16511 [21], has a systematic way of indicating the corrections by using pointers at the position of the mistake within the text with certain symbols and the type of action for corrections are placed at the left or right boundary of the line in which the corrections are made. This norm requires space to be placed at either side of the text as we write which is not common while writing with a pen. We deal with the correction which are written by placing the correction symbols and, if necessary, its corresponding corrections at a near vicinity of the place of the mistake made as shown in Figure 2.1.

2.1 Correction Types

There are different types of vocabulary corrections that are done while writing and the common types of corrections are insertion, deletion, replacement, substitution, and movement [10]. These are the corrections that we deal in this work. Each of these corrections corresponds to an action and a context for its action, which has to be determined for automatic corrections. These actions are indicated by symbols which are in one-to-one relation with them and each symbol has its own context. The context is the information that is necessary for the execution of the action, for instance, the context of the insertion symbol would be the place of insertion as well as the set of characters to be inserted.

2.2 Correction Symbols

As described in the previous section, there are basically five types of corrections. Among these five types of corrections the insertion and deletion corrections are further divided into more specific actions depending on the position of insertion and the characters upon which the insertion and deletion action would take place which increases the total number of distinct correction actions to 9. Each of these actions is represented with at least one symbol which can be seen in Table 2.1. Some actions have two symbols that represent them. This variation is due to the property of symmetry of the symbol, which are present with the “substitution” and the “insert space between characters” action symbols, and the position of the symbols, which are present with the first three insert actions in Table 2.1. Altogether we collected 13 symbols to represent the set of actions.

Two new symbols have been introduced for the actions “insert at the beginning of the word” and “insert at the end of the word” which was indicated traditional by a single insert symbol, the caret symbol, which caused ambiguity. Depending on the context of the caret symbol, the action differs. This ambiguity is shown in Figure 2.2 where the caret sign, “ ˆ ”, indicates the insertion of the

characters “is”. Humans are intelligent enough to easily disambiguate between the different semantics in such scenario using linguistic knowledge and common sense. Machines on the other hand lack these properties and have to be told in some explicit way about the actions to perform.

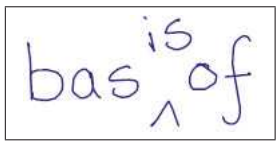
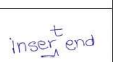
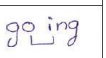
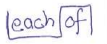
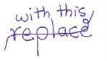


Figure 2.2: Ambiguous insert correction

The corresponding correction could be “basis of”, “bas is of”, or “bas isof” depending on where the user intended to insert the characters “is”. Among these corrections, humans are able to select the right one where as it would be difficult for machines but now with the introduction of two new symbols the machine can easily understand the action.

Table 2.1: List of Correction Symbols along with their additional information

Correction	Semantics	Symbol	Some Variation	Constraints	Example
Insertion	Insert between characters			The sides have about the same length	
	Insert at the beginning of the word			The right side has to be much longer than the left side	
	Insert at the end of the word			The left side has to be much longer than the right side	
	Insert space between characters			Lines are as straight as possible	
Deletion	Delete characters			Has to contain an intersection and the start and end has to be placed diagonally	
	Delete space			Lines are as straight as possible	
Substitution	Substitute characters			Lines are as straight as possible	
Replacement	Delete characters and replace it with some other characters			Has to contain an intersection and the start and end has to be placed diagonally	
Movement	Move characters to another destination			Starting and end has to be close to each other	

All these correction strokes can be easily drawn. Each of these symbols is drawn with a single stroke with simple and natural constraints. These constraints

are listed in Table 2.1. With these constraints, the user is free to draw the symbols as he desires. Figure 2.3 shows the deletion correction symbol that follows its constraints. It can be seen that even with these constraints, the user has the freedom to comfortably adjust to his natural writing style.

Figure 2.3: Delete character symbols following constraints having its start and end diagonally with an intersection as listed in Table 2.1 and showing flexibility in style.

Chapter 3

State of the Art

Systems that can handle corrections of free flowing handwritten text do not exist but there has been some research on editing systems using correction symbols especially on typed text documents. The system that allows digital documents to be edited on paper uses digital pen as an interface to annotate corrections. The digital documents that can be manipulated either on a computer screen or on paper are known as paper augmented digital documents (PADDs) [16]. PADDs are created as digital documents such as office documents or a CAD drawing. When a paper copy is needed, the document is printed on a special paper with pen readable patterns and a snapshot of the PADD is stored in a database. This printed document is then marked with correction symbols with the digital pen. The strokes collected by the pen are sent back to the strokes collector which will retrieve the target PADD from the database and process the pen's input. In this way the documents are edited. Figure 3.1 shows this process of editing PADDs.

Some editing systems that use paper based editing (but do not use digital pen) use scanners to transfer the information of the correction [11]. The digital document is printed on normal paper and correction symbols are marked by colored pen. The paper is scanned after the corrections. These colored marks are automatically extracted and used for editing the digital documents.

Digital pen is an interface to manage data. There are different types of devices and systems that perform similar data management. These systems are paperless systems. Many have used the idea of correcting vocabulary of digital documents in a paperless system using flat-panel displays and tablets using a stylus or a mouse [37] [25] [39] [23] [38]. With these devices the user directly edits the documents on a screen or a tablet. Some systems have buttons to indicate the correction symbols instead of drawing them and then indicating the position of the corrections using a mouse or a stylus [23].

For the purpose of correcting documents, the selection of the input device

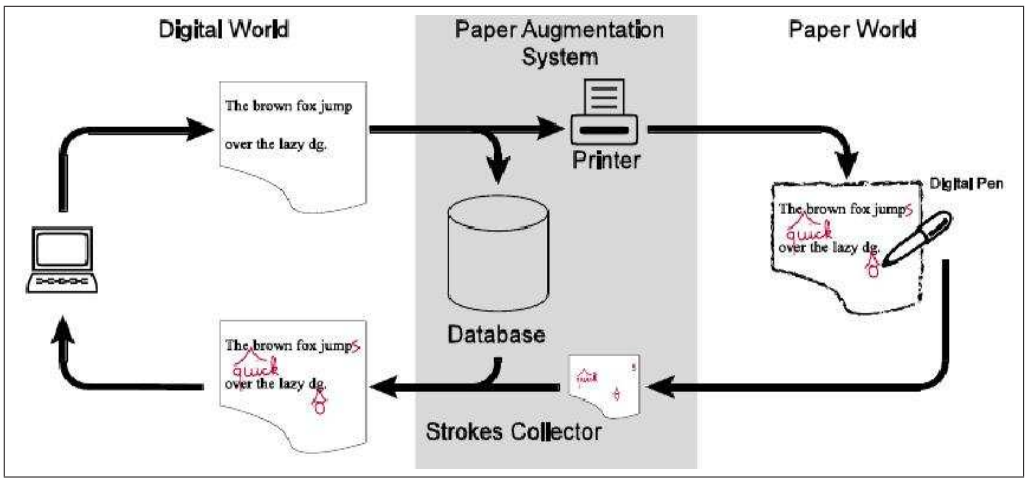


Figure 3.1: Life cycle of the PADD editing process

makes a lot of difference because of the representation of the digitized form of the written information. While using the digital pen, the stroke information is provided with the x-y coordinates and in the order in which the symbols are written with respect to the paper which differs from the information given by a stylus or a mouse which is in the form of pixel information of the screen.

Even though the input of each system is different, the recognition engine for each system is similar to any handwriting or shape recognition systems. The method used for handwriting and shape recognition is used to recognize the correction symbols. Many algorithms have been used in handwriting recognition and have been extensively studied. Among them some are template matching [9], Hidden Markov models [9] [26], neural networks [9], decision trees [25], fuzzy classifiers [29],and classifiers that use moments [44] and Fourier descriptors [44]. All of these methods have given a good result with some particular set of testing data.

Most of the recognition methods such as neural network and Hidden Markov models require a large set of training data to take into account the wide range of variants that each symbol may have. Methods that use features like moments for recognition have complex kernels and hence the complexity of the system increases. Fourier descriptors on the other hand have a relatively simple kernel but are rotational invariant which is not favorable for correction symbol recognition because the semantics of symbols are sensitive to rotation. Other methods such as template matching and decision trees are simple but rigid in representing the set of classes of symbols because of the features they use.

Each of these methods use some sort of preprocessing before the recognition is done. The general steps in preprocessing involves polynomial interpolation [7] to generate a continuous set of coordinate points for the strokes which are sam-

pled to get a set of points whose members are fixed. The sampling can be done with traditional statistical sampling or other methods such as polynomial approximation [35]. Some preprocessing also includes segmentation of each symbol for local features extraction [31] [30]. This process is a hurdle for the fast recognition of symbols which is an issue that should be eliminated without reducing the accuracy especially when the processing power is limited in a PDA.

These short comings have been eliminated in our work. The preprocessing and segmentation process has been eliminated for fast recognition of the corrections and the recognition method is simple. The algorithms for recognition use fuzzy rules. These fuzzy rules use fuzzy sets which are represented using fuzzy and crisp functions gathered using the features extracted. The idea of using fuzzy rules have been borrowed from alphanumeric recognition and mathematical expression recognition systems [31] [32] [30] [15].

These existing systems use the fuzzy and crisp functions but are local to the recognition target, which are alphanumeric or mathematical expressions, as the features are extracted on segmented parts of the target. The segmentation is shown in Figure 3.2. The method also uses preprocessing steps. The preprocessing part and the segmentation process does take extra processing and time which can be reduced by taking global features, the structural features, of the target without looking into the smaller and separate parts of the target because then the target doesn't have to be segmented and the preprocessing becomes unnecessary. Discovering global feature that will distinguish between the targets is a difficult task to do when the number of target is high. In our case we have a small set of symbols compared to the set of alphanumeric and mathematical expressions. This small size makes the extraction of global features possible. The global features take into account the information such as the distance between the starting coordinate and ending coordinates of the symbol, and the width of the symbol and so on which are global in nature to the symbol under inspection.

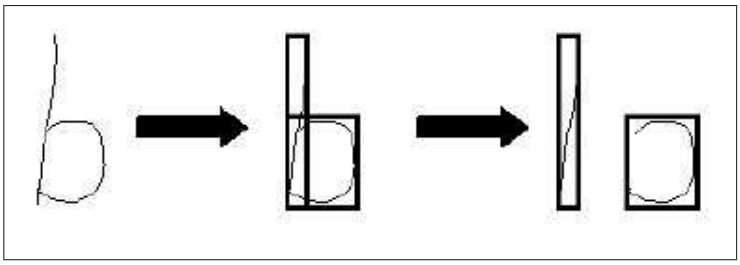


Figure 3.2: Segmentation of the character "b" to extract local features.

Like in existing systems our features are mapped onto fuzzy sets with different values assigned by the fuzzy and crisp functions. These fuzzy and crisp functions are created by analyzing the values of the features and are unique to our standard symbols. The fuzzy sets are used to build rules which are in the form of IF

ELSE rules which alone are not sufficient for classification. These rules are then implemented using three different algorithms which are decision tree, min-max algorithm, and fuzzy neural network. The implementation of the fuzzy rules has been commonly done using these methods. The fact that we use different features than other existing systems the structure of the decision tree and the FNN are different. Even with methods like these being common, a comparison between them has not been made. In this thesis we also compare the results of these different methods of classification.

Chapter 4

The Proposed System

This chapter is dedicated to the correction system. This system has three main modules: character and candidate correction stroke separator module, recognition module, and modifier module. The character strokes are the strokes that constitute the text lines whereas, the candidate correction strokes (CCS) are the strokes that constitute the correction symbol strokes and their corresponding correction strokes.

All the modules are coded in JAVA. DOM (org.w3c.DOM) package was used to manipulate the XML data files. A neural network framework, Neuroph [19] was used to create our Fuzzy Neural Network.

These modules along with the overview of the system are shown in Figure 4.1. These modules are discussed in the following sections.

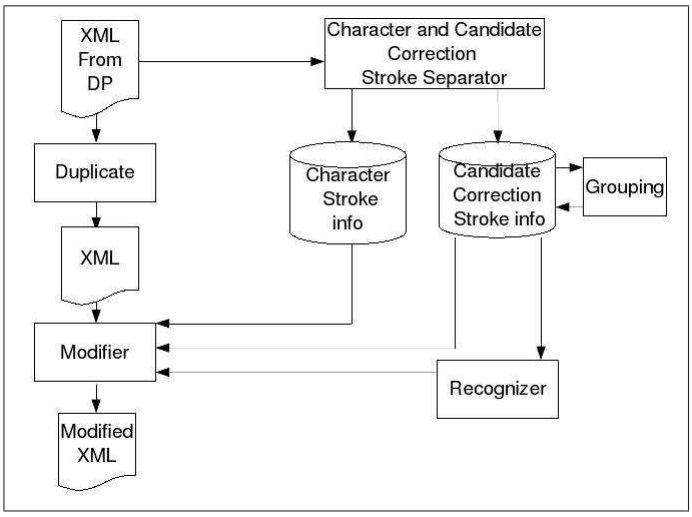


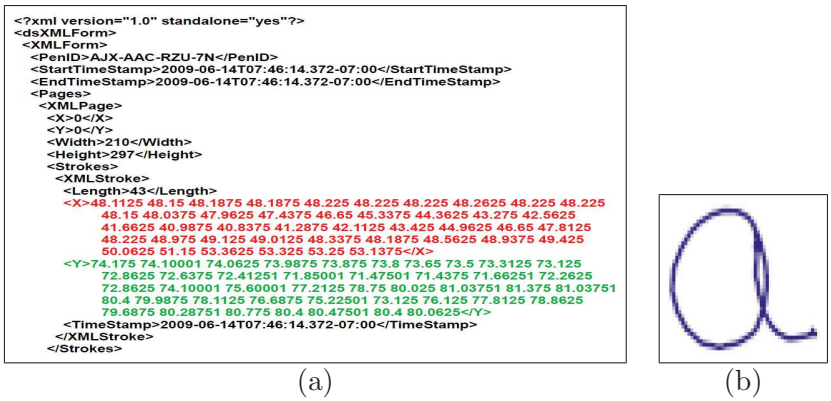
Figure 4.1: Overview of the process

This system makes some assumptions which are listed below:

1. Text written from right to left and top to bottom: strokes are not written in different text lines consecutively.
2. Two consecutive character strokes are in the same line if their y coordinate range overlaps each other.
3. The writer does not write in a cursive manner: group of characters are not written with a single stroke.
4. A gap is present between lines of texts.
5. The correction symbols are written close to the line which it corrects.
6. The correction symbol is drawn with a single stroke.
7. The segments of the correction stroke are not rewritten.
8. For each correction, the correction symbol is written first and then its corresponding correction characters (if present).

4.1 Input

The input of the system is the XML file from the digital pen. The XML file snippet along with its corresponding stroke is shown in Figure 4.2.



(a) (b)

Figure 4.2: (a) XML file snippet showing the x and y values in red and green color respectively (b) The corresponding stroke of this XML file.

This input file consists of information about the strokes written, for instance the date and time of the start and end of the writing and so on, which are placed under a node with a self explaining name as shown in Figure 4.2a. Among all the information present in this file the information that is used by the system is present in the node named XMLStroke which is the node where the information of a single stroke is present. In this node the x-y coordinate information is in the node with the tag name X and Y. The coordinates in these nodes are delimited by space.

The stroke information in the file is present in an ascending order of time in which the strokes were written. This order is exploited by the system. Without this order, managing the information would increase the complexity as we will then have to sort the XMLStroke nodes by the timestamp information present within the XMLStroke node. This order will help the detection of the characters strokes and CCS as there is no explicit distinction between them in this file. Such distinctions are required so that only the CCS is passed on to the recognition module. This separation is done by the character and CCS separator module.

4.2 Separation

The character and correction stroke separator module takes the XML file as its input. The stroke information in this file is separated into the character strokes and the CCS. This helps in the recognition process as it reduces the classes of strokes to recognize, making this step an important module for the correction system. This separation is essential in every correction system. In existing correction systems, this separation is not explicitly done because these data are not merged together in one file. The digital documents are stored in one file and the CCSs are stored in a different file.

This separation module stores the character strokes and the CCS in separate lists. With this separation, future operations can be carried out efficiently. Along with the separation, each stoke is associated with some unique line of text. This will indicate which line of text is composed of character strokes and what are the CCSs associated with that line. This information will help the incorporation process of the corrections in the modifying module. The following two sections indicate how the strokes are separated and are assigned to a line of text.

4.2.1 Character Stroke

In the pool of strokes in the XML file, clustering strokes in text lines is an interesting and challenging task. There have been methods to extract text lines using dynamic programming [28]. This method uses partial histograms which are formed from images which makes this method unsuitable for our purpose because the image in our case is formed at the end of our process.

Detecting the character stroke is a systematic process in which we first start by taking the first stroke which is the first set of x and y coordinates in the XML file. The first stroke written is always assumed to be a character stroke. The next character stroke could be a character stroke on the same or a different text line. To determine if the next written stroke is a character stroke, we check its position with respect to the previous stroke and check if their y coordinates range overlaps. If the current stroke is on the right side of the previous character stroke and an overlap of their y coordinate range exist then the current stroke is a character

stroke as shown in Figure 4.3. This position and overlapping property identifies the character stroke and also indicates that both the previous and the current character stroke are on the same line of text. If the current stroke is below the previous character stroke with a distance greater than a fix threshold value set for our purpose then the current stroke is a character stroke but with a different line number. This method of threshold is used by existing systems to detect the characters in a text line [26] [36]. Once the character stroke is detected, it is placed on a list with additional information of the text line it is present in.

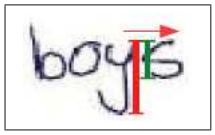
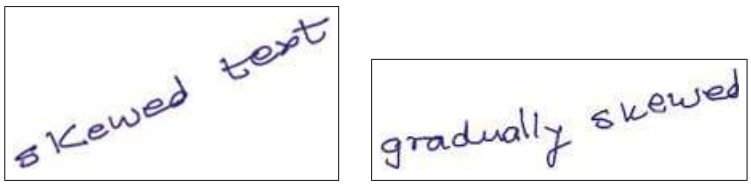


Figure 4.3: "s" is the current stroke which is on the right side of the previous character stroke "y" indicated by the horizontal arrow and their y coordinate range overlaps each other indicated by the vertical lines which helps us determine the current stroke "s" is a character stroke.

The method of overlapping works well except for the case of the "dot" in the characters "i" and "j" because the current and the next strokes y coordinate range does not overlap each other. In these cases the dots are ignored. This overlapping method of detecting strokes of a text line can also handle slightly skewed text lines and gradually skewed text lines until consecutive character strokes overlap their y coordinate range(see in Figure 4.4).



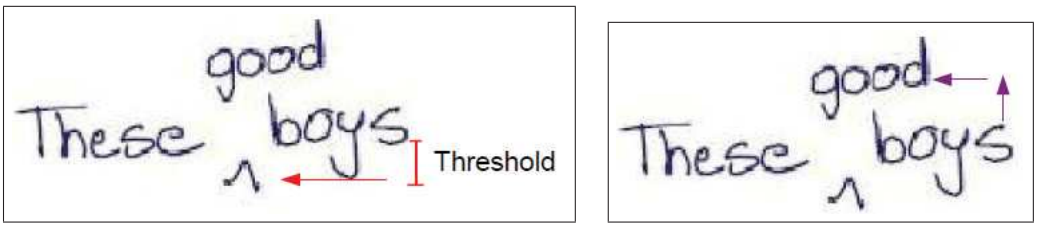
(a) (b)

Figure 4.4: (a)Slightly skewed and (b)Gradually skewed text which can be identified as a line using the overlapping method.

4.2.2 Candidate Correction Stroke

The concept of the position of the stroke and overlapping exists in detecting the CCS as in detecting the character stroke. There are three possible cases in which a CCS could be detected. If a stroke is positioned on the left side of the previous character stroke and the y coordinate range of these two stroke overlap then the stroke is a CCS. The second case is when a stroke is above the previous character stroke and the third case is when a stroke is below the previous character stroke with a distance less than a pre assigned threshold value (see Figure 4.5a and b).

The information of these strokes is stored separately from the text line strokes. These strokes are only candidates for the correction symbols. A CCS could be a correction symbol or a correction character of some correction symbol as mentioned earlier at the beginning of this chapter.



(a) (b)

Figure 4.5: (a) CCS present below the last character stroke with a distance not greater than a fix threshold (b) CCS detected when a stroke is above and on the left of the last character stroke.

The CCS is associated with a particular line which is determined by measuring the distance between the CCS and some character stroke. The character stroke that has the smallest distance with the CCS is the text line the CCS is associated with. This association is necessary to determine the text line on which the corrections will be made. The associated text line will contain a stroke that is always closest to the CCS among all the other strokes present on other text lines as shown in Figure 4.6. The distance between two strokes is measured from their center of gravity.

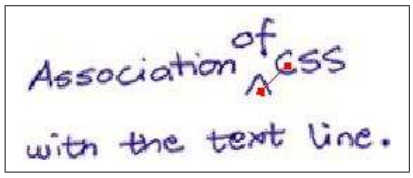


Figure 4.6: The red line indicates the distance which is the smallest among all the possible distances.

While incorporating the corrections, the correction symbol indicates what actions to take and whether there are correction texts to be added to the text line. The correction characters have the property of the text line as they are written in a manner such that the y coordinates of two consecutive correction character strokes overlap each other. The same principle of grouping is followed to group the candidate correction symbol with an additional condition. This condition is that the correction texts should be close to each other. This closeness is determined by selecting a threshold value. CCSs from the CCS list are then grouped together by these principles. This grouping is done so that when an operation is done on one of the group members, the same action is taken with all

the following members. This simplifies and in turn speeds up the incorporation process.

4.3 Symbol Recognition

The recognition module classifies the correction symbols provided by the character and candidate correction stroke separation module. Features that are extracted from these symbols are used to recognize them. The extracted features from the symbols are mapped on to different fuzzy sets using fuzzy and crisp functions. These fuzzy sets are used in fuzzy rules implemented using three algorithms which are: decision tree, min-max, and fuzzy neural network. This recognition module uses the advantages of fuzzy set theory.

4.3.1 Fuzzy Logic

Fuzzy logic (FL) was conceived by L. Zadeh [43]. It is a way of processing data by allowing partial set membership rather than crisp set membership or non-membership. It has been extensively used in control systems [34]. Two valued logic systems consider functions whose values are among the set true or false usually represented by 0 and 1 respectively. In contrast, fuzzy logic considers membership functions or fuzzy functions whose truth values are fuzzy set of the unit interval. A fuzzy set is characterized by a membership function mapping the elements of a domain, space, or universe of discourse X to the unit interval 0 and 1 [43]. That is,

$$\text{A: X} \rightarrow [0, 1].$$

In principle, any function of this form describes a fuzzy function associated with a fuzzy set A that depends not only on the concept to be represented, but also on the context in which it is used. The function graphs vary depending on the functions. Some of the shapes that the function might take are, triangular, bell, trapezoidal, haversine, or exponential. Figure 4.7 shows three trapezoidal fuzzy functions corresponding to its fuzzy sets cold, warm and hot. The fuzzy functions take in the temperature value and returns values in the range of 0 and 1 which represents the membership of the temperature value in the fuzzy sets of cold, warm and hot. A point on the temperature scale has three truth values, each corresponding to a fuzzy set. This process of categorizing the temperature value into cold, warm and hot is called fuzzfication process because the classification among the three set is not crisp. Crisp functions are not suitable for such representation of the degree of cold, warm or hot because the functions tend to be more rigid hence unnatural [5]. For example, a temperature of 25 degree Celsius could be warm and hot depending on the place and hence cannot be classified as a certain category of warm or hot.

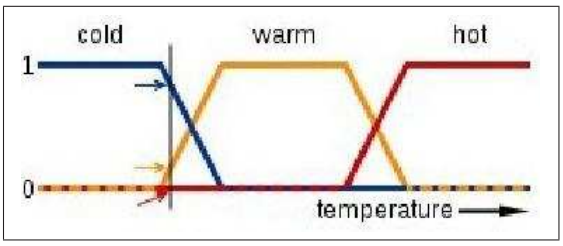


Figure 4.7: Trapezoidal fuzzy functions for cold, warm, and hot.

The value for each fuzzy function is in the range of 0 to 1 similar to the probability measure range, but it does not represent the probability. It rather gives a sense of the possibility measure.

4.3.2 Feature Extraction

Recognition is done on the basis of features that are extracted from the symbols. Due to the elimination of the preprocessing step the features extraction has to be done carefully and should be specific towards the symbol. The features are selected in such a way that it has certain properties that represent the shape of the symbol for example, the distance between the start and the end of the stroke indicates whether the stroke is circle or not aiding to the recognition of the movement correction symbol.

A fuzzy function helps to formalize the expressive power of a feature. Each feature gives a value which cannot express them in terms of categorizing the symbols. Fuzzy functions elegantly represent the set of symbols that the feature tends to represent. Some features represent linguistic values such as "almost equal", "longer than" or "shorter than" which cannot be expressed by crisp functions and some features such as "is there an intersection" is better expressed with crisp functions so, depending on the features, fuzzy functions and crisp functions are used to represent the fuzzy sets.

There are 10 fuzzy functions which use 4 features and 6 crisp functions which use 6 features. The 4 features that the fuzzy functions use are described below:

Feature 1: This feature is the ratio between the distance between the start and end point of the symbol and the distance between the center of gravity of the symbol to the line joining the start and end point of the symbol. These operands of the ratio are shown in Figure 4.8a. Two fuzzy functions are used to categorize the complete set of symbols in two different sets. The fuzzy functions and sets are graphically represented in Figure 4.8b and c.

Feature 2: This feature is the ratio between the distance of the start and end point and the width of the symbol which can be visualized from Figure 4.9a. Two fuzzy functions whose graphical representation is in figure 13.b are aimed to categorize the fuzzy set 1 produced using feature 1 to two smaller fuzzy sets.

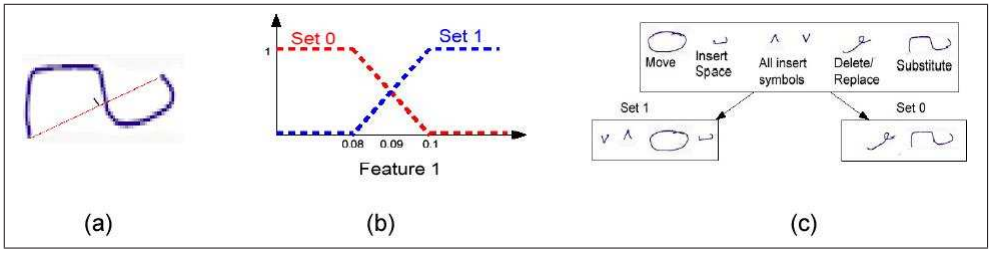


Figure 4.8: For feature 1:(a)The operands of the features are in red and black. (b)The fuzzy functions (c)Members of the fuzzy sets.

The member of the fuzzy sets can be seen in Figure 4.9c.

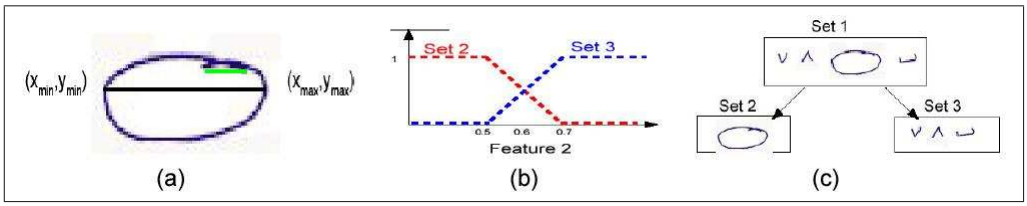


Figure 4.9: For feature 2:(a) The operands for the feature are in green and black. (b) The fuzzy functions (c) Members of the fuzzy sets.

Feature 3: This feature is the ratio between the length of the symbol from the start to end and the length of the segment between the start and the highest point of the symbol which are shown in Figure 4.10a. Three fuzzy functions are used to categorize the upward insert symbols into insert left, insert right and simple insert. The graphical representation of these functions is shown in Figure 4.10b. Figure 4.10c shows the members of the fuzzy sets.

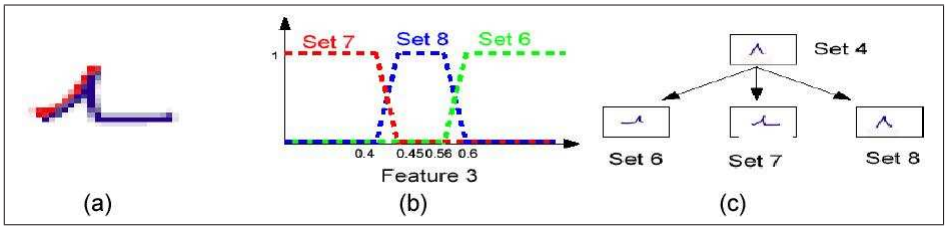


Figure 4.10: For feature 3:(a) The operands of the feature are in red and blue. (b) The fuzzy functions (c) Members of the fuzzy sets.

Feature 4: This feature is similar to feature 3. It is the ratio between the length of the symbol from the start to end and the length of the segment between the start and the lowest point of the symbol which are shown in Figure 4.11a. These fuzzy functions are aimed to categorize the downward insert symbols into insert left, insert right and simple insert sets. The functions are graphically shown in Figure 4.11b. Figure 4.11c shows the members of the fuzzy sets.

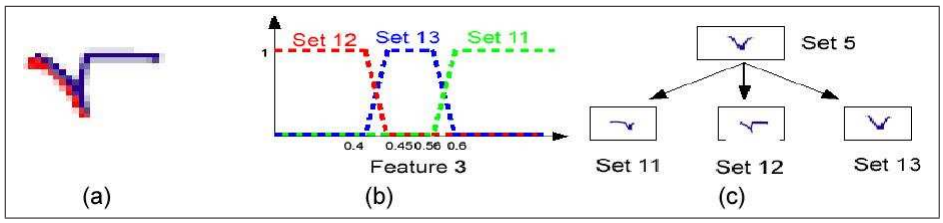


Figure 4.11: For feature 4:(a) The operands for the feature are in red and blue. (b) The fuzzy functions (c) Members of the fuzzy sets.

The features that are used by the crisp functions are explained below:

Feature 5 and 6: These features indicate the direction of the insert and delete space symbols and they are represented by crisp functions. The insert symbols and the delete space symbols are separated in 2 sets, according to the upwards and downwards direction as shown in Figure 4.12. Two crisp functions are used to indicate this direction. The angle between the point which is farthest from the line joining the start and the end point and the horizontal axis is measured. If the furthest point is above the horizontal axis the angle it makes will be between 0° to 270° indicating the symbol is directed upwards else it would be directed downwards. Whenever a direction is detected, the output of the corresponding crisp function is set to 1 and the other to 0.

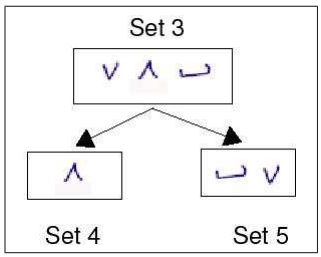


Figure 4.12: Members of the fuzzy set created by the crisp functions

Features 7, 8 and 9: represent the structure of the correction symbol with the vertical and horizontal segments in the strokes. The vertical segment is represented by V and the horizontal is represented by H. We are concerned with only 3 patterns which are VHV, VHVHV and HVH indicating the strokes corresponding to delete space, substitute, and insert space respectively as seen in Figure 4.13. Each pattern is represented by a crisp function. The horizontal and vertical concepts are created using the accumulation of the vertical and horizontal property of the segments of the strokes. The vertical and horizontal property is known with the segment slope. If the slope is greater or equal to one then the segment is horizontal else it is vertical. The smallest segment of the stroke joining consecutive points is checked if they are V or H. The patterns with V and

H are formed by recording the change in the orientation of the segments which are significantly long. If one of the three patterns is detected then the function that is responsible for the pattern returns a value 1 and the rest returns 0.

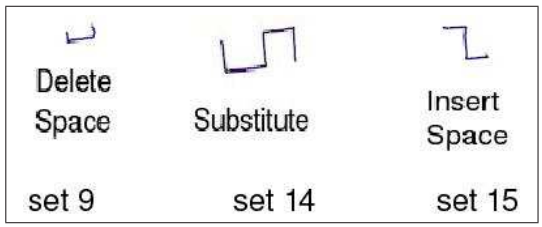


Figure 4.13: Members of the fuzzy set 9, 14 and 15

Feature 10: This feature is also represented by the crisp function. The function indicates if the correction symbol has segments that intersect each other. In this case, the function returns 1 otherwise 0.

Figure 4.14: Members of the fuzzy set 10

4.3.3 Fuzzy Rules

Classification of the symbol is done by fuzzy rules. These rules are used to determine the most likely symbol that a stroke represents by the fuzzy sets. These rules state the attributes and relationships that a fuzzy set has with the correction symbol. Each rule is handcrafted and made for a symbol. The rules can be interpreted as IF THEN rules. For example, let's take the character "b" and segment it in such a way we get the structure "I" and "⊃" making these properties its feature. These two features include the character "b" in two sets, the sets of "has structure I" and "has a structure ⊃ at the bottom", a fuzzy rule could be made by the following proposition, IF "contains a straight line" and "has a circle at the bottom" THEN "the character is b". Similarly, fuzzy rules for the correction symbols are listed in the Table 4.1 in terms of the sets created by the fuzzy and crisp functions.

The fuzzy function allows each symbol to be in every set with a value between 0 to 1. The fuzzy rules listed in Table 4.1 can be used for classifying stokes in correction symbols. The property of fuzzy functions allows more than one fuzzy rule to be triggered when classification is done using these fuzzy rules stated in

Table 4.1. This problem is solved by using methods that are capable of selecting the best rule. We use decision tree, min-max algorithm and fuzzy neural network for this purpose.

Table 4.1: Fuzzy rules for Correction Symbols

IF	THEN
in set 0 and set 10	delete
in set 0 and set 14	substitute
in set 0 and set 15	insertspace
in set 1 and set 2	move
in set 1 set 3 set 5 and set 9	delete space
in set 1 set 3 set 4 set 6	upwards left insert
in set 1 set 3 set 4 set 7	upwards right insert
in set 1 set 3 set 4 set 8	upwards insert
in set 1 set 3 set 5 set 11	downwards left insert
in set 1 set 3 set 5 set 12	Downwards right insert
in set 1 set 3 set 5 set 13	Downwards insert

4.3.4 Decision Tree

Decision trees have been defined as a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility [2] [41] [33]. Decision trees are used for decision analysis, to get the most possible goal. The fuzzy rules listed in Table 4.1 are modified to achieve unique classification of the correction symbol. The decision tree is shown in Figure 4.15.

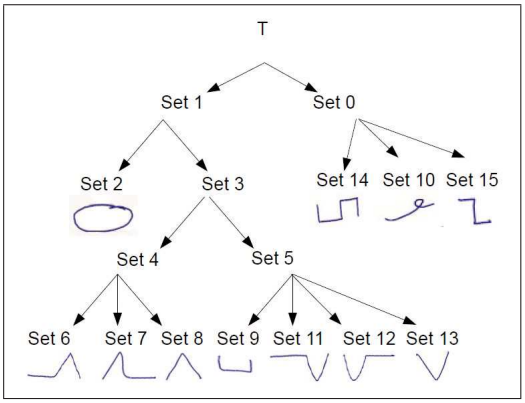


Figure 4.15: Decision tree with the symbols at the leaves.

In the decision tree, each leaf consists of a single correction symbol. Each node in the tree represents a set. The parsing is done from the root to the leaf.

Decision is taken at each node about the branch to follow. This decision is done by comparing the values for each sibling of the node. The sibling that has the highest value is the branch that is followed. The decision can be represented in the form of IF THEN rules like in the fuzzy rules with some slight modification. The modification can be seen in the following example for the rule of the delete space symbol:

IF		$value(set1) > value(set0)$
	AND	$value(set3) > value(set2)$
	AND	$value(set5) > value(set4)$
	AND	$value(set9) > value(set11)$
	AND	$value(set9) > value(set12)$
	AND	$value(set9) > value(set13)$
THEN		"delete space symbol"

Even though this method insures that only one rule is triggered for a single stoke, while making every decision at the nodes it selects one single path which removes the chances of selecting all the symbols that are present on the other path. This strong decision may cause problems in the top level of the tree when the values that are compared differ only slightly. This strong decision problem is removed with the min-max algorithm.

4.3.5 Min-Max Algorithm

This method is used in decision theory, game theory, statistics and philosophy for minimizing the maximum possible loss [6] [40]. The fuzzy rules are composed of sets in its conditions and each set is represented by a value from the fuzzy or crisp function. Min-Max algorithm selects a rule that has the maximum of the minimum values among the sets in the rules.

Min-max algorithm can be seen as the generalization of the decision tree. Figure 4.16 shows a decision tree with rules shown as boxes.

The boxes engulf the sets that constitute a rule. The red box indicates that the move symbol rule constitutes set 1 and set 2. The green box represents the sets 1, 3, 5, and 9 which consists the fuzzy rule for delete space. The min-max algorithm selects the rule that has the maximum value among the minimum values of the sets in each box. This method gives priority to all the set values equally unlike the decision tree where the decision is taken at each node to eliminate the symbols that are present in the unselected path. In this method we have the list of minimum values that represent the rule, which can be used for ranking purposes and to provide alternative symbols other than the maximum one if necessary. In this work we take the maximum out of this list of minimums.

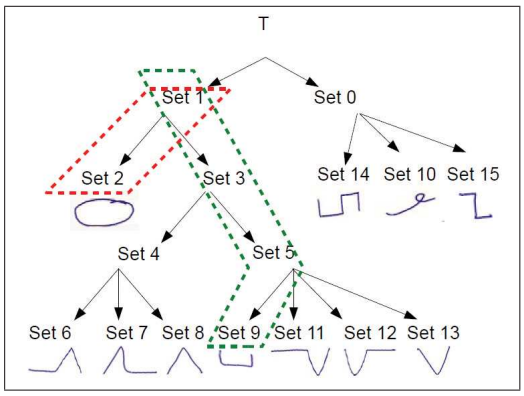


Figure 4.16: The box represents the sets that forms the rules.

4.3.6 Fuzzy Neural Network (FNN)

It is the combination of neural networks (NN) and fuzzy logic [12]. A neural network has been defined as "an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns" [17].

The NN are categorized according to these connections. An example of a neural network is shown in Figure 4.17.The first layer of the network is the input layer that receives the input, shown as the vector x. The number of neurons in this layer is the same as the number of inputs. The last layer is the output layer which gives the output, shown as the vector c. In the case of classification, each neuron in the output layer corresponds to a class and normally when an input is classified as a class the corresponding neuron has an output of 1 whereas the rest are 0.

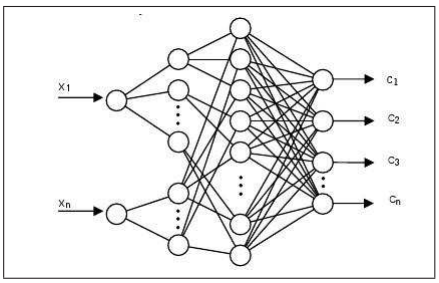


Figure 4.17: A Neural Network.

The NN is trained to estimate the weights of the connections. There are different training processes among which we will be using the Widrow-Hoff rule also known as the LMS rule to train the FNN [42]. This is a supervised training

method with the following equation to estimate the weights:

$$w = w^{old} + \frac{BEx}{|x|^2}$$

In this expression, B is learning constant between 0 and 1, E is the error computed between the actual output and the output of the network, x is the input vector and w is the weight vector [17].

Fuzzy neural network are certain types of neural networks, which are distinguished by its connections and has been used to implement the fuzzy rules in Table 4.1. The network shown in Figure 4.18, is a partial structure of the FNN to understanding the structure of the network.

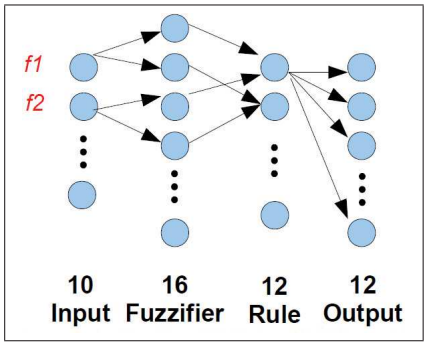


Figure 4.18: Partial structure of the FNN.

The FNN has four layers. The input layer has 10 neurons each for the 10 features. The neurons in this layer have a linear transfer function which passes on whatever the input is to the second layer. The second layer is the fuzzification step corresponding to the generation of fuzzy sets. This layer has 16 neurons. The transfer function of the neurons is the fuzzy and crisp functions. The connection between the first and the second layer is in a way that the feature that is in the neuron of the input layer is connected to the neurons in the second layer which has the transfer function associated to it as defined in the feature extraction and fuzzy set section. The third layer is the rule layer where each neuron represents a fuzzy rule as in Table 4.1. The neurons in this layer are connected with the neurons in the second layer which represents the sets that consists the rule. The transfer function is a linear function. The last layer is the output layer. Each neuron corresponds to a symbol in the leaf node of the decision tree shown in Figure 4.15 and with an extra stroke which corresponds to a straight line stroke of the character “F”,“I” and “T”. This straight line is required in FNN because these are also separated as candidate correction stroke and if this is not recognized, then it will be recognized as some other correction symbol which will ruin the system therefore it has to be detected. Hence, layer consists of 12 neurons which have the step transfer function. This layer is fully connected with the rule

layer. This network is trained with the LMS rule to estimate the weights of the output layer. 130 samples, which consists of 10 samples per symbol, were used to train the network. The network converged in 5 iterations with zero error. The training is diverged only if contradictory samples are present.

4.4 Context Extraction

Recognition begins the correction incorporating process. After the corrections are recognized the implementation of its meaning, i.e. delete, requires the context upon which correction actions will be used. Context extraction is the term used to extract information which is relevant to the correction symbols. For example, the delete symbol would require the characters that have to be deleted and for the insert symbol it would require the characters to be inserted as well as the position to be inserted. The context and the method used in the system to extract this information are explained in this section.

Delete/Replace Characters

The delete correction symbol expresses the characters to be deleted by the property of intersection. Our system detects the starting and ending characters that the delete symbol intersects and deletes the range of characters. This intersection is essential for correction system. Humans do not need the intersection property of the starting and ending characters as shown in Figure 4.19.

Figure 4.19: Detection of delete word using overlapping property by humans

Here the x coordinate overlapping property is enough to indicate the deletion. This is not possible for computers as there may could be situations as shown in Figure 4.20, in which the overlapping exists and yet do not mean the deletion.

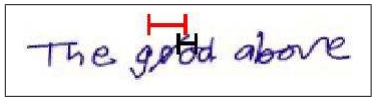


Figure 4.20: Property of the x coordinate overlap is not enough.

This deletion symbol also can be made to express the replacement action. If there is some correction characters present above the deletion symbol and are associated with the same line then the deletion symbol represents the replacement meaning. The context for the replacement symbol is the same as the deletion symbol with the addition of the correction symbols present above it.

Delete Space

The delete space indicates the deletion of the space between two characters. These characters have the property of having the minimum distance to the start and end point of the symbol. This can be seen in Figure 4.21 with the red lines indicating the shortest distance. This property is measured and the context is recorded for the incorporation process.

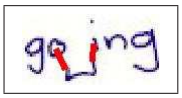


Figure 4.21: Character closest to the start and end indicates the deletion of space between them.

Insert

Insertion of characters is done between two characters. These two characters will be the closest to the highest or the lowest point of the insert symbol depending on direction of the symbol. Figure 4.22 illustrates this point with the red lines indicating the shortest distance.

Figure 4.22: Two characters which are closest to the insert symbol.

This distance is measured to extract the context. The other part of the context is the characters that have to be inserted at that point. The correction symbol that is written immediately after the insertion symbol is the inserting character. Any other consecutive characters that are in the same group as the correction character are also the characters for insertion.

Substitute

This symbol indicates the group of symbols to be substituted in the text. The context is extracted by the concept of intersection of the vertical segments of the symbol by the character residing on its sides. This is illustrated in Figure 4.23.

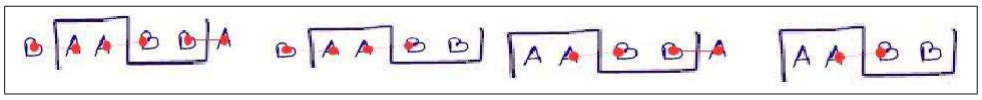


Figure 4.23: Different intersecting possibilities.

Depending on the position of the correction symbol, the number of intersection varies. This information is vital during the extraction of the context. The different intersecting possibility is illustrated in Figure 4.23. Once the intersecting characters are known the range of characters is recorded as the context of the correction symbol.

Move

The move correction symbol circles it's context to indicate the characters to be moved. This context is extracted by checking which character has its X axis range within the range of the delete symbol. The x coordinate range is shown in Figure 4.24 with the red line. The move symbol uses a line that shows where these characters have to be moved. One end of the line intersects this correction symbol and the other end of the line indicates where the symbol has to be inserted. The movement is done before or after a character. This information is extracted and stored for the process of incorporation.

Figure 4.24: The overlapping of the X coordinate axis shown by the red line of the move symbol with it's context characters.

4.5 Incorporation

The distinction between the correction symbol and its correction character is defined by the correction symbol's semantics. The incorporation process is a systematic process in which the first CCS in the CCS list will always be a correction symbol. When the correction symbol is recognized, its semantics indicates whether the correction strokes are present or not. For example, the insert symbol indicates that there must be correction strokes whereas, the insert space symbol doesn't require any correction strokes. This helps in filtering out the correction symbols from the list. When a correction stroke is required, the stroke which follows the correction symbol is the beginning of its correction stroke. The grouping comes in handy here. The strokes present in the group of the first correction stroke are the set of correction strokes for the correction symbol. This separates the correction symbol with the correction strokes. The correction symbols are selected and passed on to the recognition module.

Incorporation is the step in which the corrections indicated by the correction symbols are made. This step requires the recognized symbol and its context to operate. The implementation of the corrections involves the manipulation of

the x-y coordinates of the character strokes and sometimes even removal of the character stroke information from the XML file. These changes are made on a duplicate file of the original file to retain the original information. Every symbol represents some unique action to be taken. These actions are done on each line of text separately. And after modifying all the text lines, if any strokes are placed outside the page limit the strokes are moved below to the next line for formatting purpose. The actions for incorporation are described and illustrated below.

Insertion

The context of this symbol consists of the character symbol and the correction characters are to be inserted which are inserted between them. The x-y coordinates of the characters that have to be inserted are changed so that the starting character of the inserted character is the same as the character before which it is being inserted. This character is then shifted behind the inserted character with the same distance it had with the character after which the insertion was done. This process is shown below in Figure 4.25.

Figure 4.25: Insertion.

An example with the process of insertion follows: To incorporate the insertion correction, a caret symbol,“ ˆ ”, is placed at the position of insertion. Figure 4.26 below shows an insertion correction scenario where the user wants to insert a character “b” between the characters “a” and “c”. The sequence in which the characters were written is: “ a ”,“ c ”,“ ˆ ”, and,“ b ”.

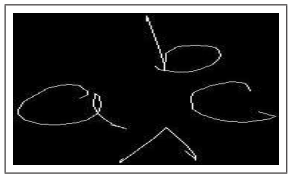


Figure 4.26: The written text to illustrate the insertion correction symbol.

Corresponding to Figure 4.26 the snippet of the XML file generated is shown in Figure 4.27. The x-y coordinates of the character “a” is in green, “c” is in blue, “ ˆ ” is in red, and “b” is in orange. The order in which the stroke information placed in the XML file indicates the sequence in which the strokes were written.

The strokes “a” and “c” are detected as the text line stroke because of the overlapping of their y range, and character “c” has a greater x coordinates than

“a”. The “ ^ ” and “b” are detected as candidate correction strokes. This is because these strokes have their x axis values less than the character “c” and are close to “c”. The correction symbol is always written first and then its correction character, this entails that the first candidate correction stroke is always a correction symbol so; this symbol is sent to the recognizer which correctly recognizes it. After recognizing the insertion symbol, we identify its correction character because the insertion symbol requires at least one correction character. The correction character for insertion is the immediate candidate correction stroke which is identified for incorporation.

After recognizing the stroke and identifying the character to be inserted, the incorporation module incorporates the character “b” between the character “a” and “c” by changing the x-y coordinates of the correction character. The y coordinates increases as we go down the paper. Therefore, the y coordinates of ”b” has to be increased to place it between “a” and “c”. The increase is done by adding the distance between the difference of the y coordinate of the center of gravity, mid point of the x-y coordinates, between the characters “a” and “b” to the y coordinates of “b”. The x coordinate of “b” is modified in such a way that the gap between the characters “a” and “c” is preserved between “a” and “b”, and also between “b” and “c”. With the movement of “b”, all the characters which are after “a” must change their x coordinate. After the modification, the correction symbol is removed from the XML file.

The resulting strokes are shown in Figure 4.28 and its corresponding snippet of the XML file is shown in Figure 4.29. The color of the x-y coordinates represent the same characters as in the Figure 4.27.

Here is another example where the insertion is not done correctly:

Figure 4.30 shows the strokes drawn in the sequence of “ a ”,“ c ”,“^ ”,“ b ”,“ g ”. The input XML file is given in Figure 4.32.

Each color indicates a stroke in the input where blue is for “a”, green for “c”, red for “ ^ ”, yellow for “b” and orange for “g”. The detection of the character strokes “a” and “c” and of the candidate correction symbols “ ^ ” and “b” are done in the same way as in the previous example. The new stroke “g” is classified as the character stroke because its y coordinate range overlaps with the character “c” hence when incorporation of the correction is done “g” is not inserted in the position. The incorporation is done in the same way as explained in the previous example. The stroke “g” is then shifted to its right in the same distance as “c” because it is treated as the character stroke. The output is given in the Figure 4.32 and the output xml file is shown in Figure 4.33 . The color in this stroke indicates the same strokes as in Figure 4.31.

```
<Strokes>
        <XMLStroke>
        <Length>29</Length>
        <X>126.9 127.0875 127.425 127.425 127.275 127.0875 126.9 126.225 125.6625 124.725 123.1875
                122.7375 122.7 123.15 124.275 125.475 126.6 127.425 128.1375 128.325 128.25 127.95
                127.8375 128.1375 128.5125 129.2625 130.0125 129.9375 129.8625</X>
        <Y>31.9125 31.8375 31.425 30.8625 30.6375 30.3375 30.075 29.8875 29.925 30.45 32.4375 33.75
                34.3125 35.2875 35.925 36.0375 35.7 35.2125 34.2 33.1875 31.6875 31.2 33 34.3875 35.175
                36.075 36.6375 36.525 36.3</Y>
        <TimeStamp>2009-06-14T07:44:35.226-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>18</Length>
        <X>138.45 138.2625 138.1125 137.85 137.1 135.7875 134.6625 134.4 134.55 135.15 136.0125
                137.0625 138.1125 139.725 140.1375 140.025 139.95 139.35</X>
        <Y>30.825 29.925 29.7 29.5125 29.475 30 31.05 31.8 33.675 34.4625 35.1 35.5125 35.625 35.1 34.72
                34.6875 34.65 34.425</Y>
        <TimeStamp>2009-06-14T07:44:36.962-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>14</Length>
        <X>129.825 129.6 129.6 129.6 129.6375 132.15 132.7875 134.2875 134.7 134.7375 134.7375
                134.775 134.625</X>
        <Y>41.2875 41.5125 41.5875 41.775 41.85 41.8125 37.725 36.4125 39.4875 40.8 41.4 41.4 41.2125
                40.575</Y>
        <TimeStamp>2009-06-14T07:44:37.788-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>21</Length>
        <X>131.4375 131.4375 131.4 132.3 132.6 132.6375 132.75 133.575 134.775 135.4125 136.05 136.35
                136.425 136.2375 135.825 134.775 134.1 133.4625 132.45 132.2625 132.225</X>
        <Y>20.1375 19.35 19.35 24.75 26.85 27.7875 25.05 24.075 23.8125 23.925 24.1875 24.7125 25.425
                26.1375 26.8875 27.7125 27.9375 27.9 27.6375 27.525 27.45</Y>
<TimeStamp>2009-06-14T07:44:39.247-07:00</TimeStamp>
</XMLStroke>
</Strokes>
```


Figure 4.27: XML file snippet with color to distinguish the different strokes. Green for “a”,blue for “c”, red for “^”,and orange for “b”.

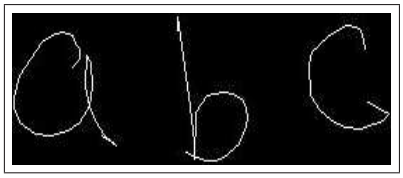


Figure 4.28: The strokes after the incorporation of the corrections

Deletion/Replacement

Depending on the context, the deletion symbol can represent a delete action or a replace action. If it is a delete action the characters that are present in its context is removed from the XML file and the characters after the last deleted character are shifted to the position where the first character was deleted. If the symbol has its semantics as a replacement symbol then the characters that are intersected are removed from the XML file and the correction character strokes are moved to the place of the deleted characters. The characters which replace the deleted character maintain the same distance as the replaced characters. This deletion and replace operation is shown in Figure 4.34.

```
<Strokes>
        <XMLStroke>
        <Length>29</Length>
        <X>126.9 127.0875 127.425 127.425 127.275 127.0875 126.9 126.225 125.6625 124.725 123.1875
                122.7375 122.7 123.15 124.275 125.475 126.6 127.425 128.1375 128.325 128.25 127.95
                127.8375 128.1375 128.5125 129.2625 130.0125 129.9375 129.8625</X>
        <Y>31.9125 31.8375 31.425 30.8625 30.6375 30.3375 30.075 29.8875 29.925 30.45 32.4375 33.75
                34.3125 35.2875 35.925 36.0375 35.7 35.2125 34.2 33.1875 31.6875 31.2 33 34.3875 35.175
                36.075 36.6375 36.525 36.3</Y>
        <TimeStamp>2009-06-14T07:44:35.226-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>18</Length>
        <X>147.8625061035156 147.6750061035156 147.52500610351564 147.26250610351562
                146.51250610351562 145.20000610351562 144.07500610351562 143.81250610351563
                143.96250610351564 144.56250610351563 145.4250061035156 146.47500610351562
                147.52500610351564 149.13750610351562 149.5500061035156 149.43750610351563
                149.3625061035156 148.76250610351562</X>
        <Y>30.825 29.925 29.7 29.5125 29.475 30 31.05 31.8 33.675 34.4625 35.1 35.5125 35.625 35.1 34.725
                34.6875 34.65 34.425</Y>
        <TimeStamp>2009-06-14T07:44:36.962-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>21</Length>
        <X>134.4375 134.4375 134.4 135.3 135.6 135.6375 135.75 136.575 137.775 138.4125 139.05 139.35
                139.425 139.2375 138.825 137.775 137.1 136.4625 135.45 135.2625 135.225</X>
        <Y>29.756251525878906 28.968751525878908 28.968751525878908 34.368751525878906
                36.46875152587891 37.40625152587891 34.6687515258789 33.69375152587891
                33.431251525878906 33.5437515258789 33.806251525878906 34.331251525878905
                35.0437515258789 35.75625152587891 36.50625152587891 37.331251525878905
                37.556251525878906 37.518751525878905 37.25625152587891 37.143751525878905
                37.06875152587891</Y>
        <TimeStamp>2009-06-14T07:44:39.247-07:00</TimeStamp>
        </XMLStroke>
</Strokes>
```


Figure 4.29: Modified XML file snippet with color to distinguish the different strokes. Green for “a”, blue for “c”,and orange for “b”.

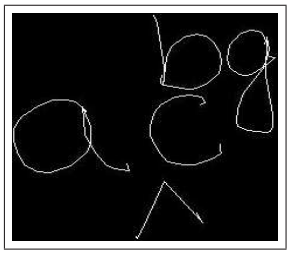


Figure 4.30: The written text to illustrate the insertion correction symbol.

Substitution

The context of the substitute symbol provides the two set of characters to be substituted. One set comes before the other. The set that comes second is shifted in the place of the first set with the x coordinates of the starting character. The first set with the lesser x valued coordinates are shifted behind the second set with the same gap that existed before the shifting operation. This substitution is shown in Figure 4.35.

```
<Strokes>
        <XMLStroke>
        <Length>33</Length>
        <X>142.275 142.4625 142.4625 142.5 142.5 142.5 142.5 142.5 142.425 141.6 140.925
                139.7625 138.45 137.0625 136.65 136.3875 136.3125 136.725 137.2875 138.3 139.425
                141.225 142.65 142.95 142.9875 142.2375 142.4625 143.1 143.4 144.2625 145.0875
                146.2125 146.0625</X>
        <Y>60.75 60.675 60.7125 60.7125 60.7125 60.7125 60.675 60.6375 60.525 59.7375 59.55 59.5875
                60.1875 61.2375 61.8 62.5125 63.7125 65.175 66.1875 66.8625 66.9375 66.2625 64.8375
                63.9375 62.2125 60.2625 63 64.575 65.0625 66.1125 66.525 66.75 66.75</Y>
        <TimeStamp>2009-06-15T18:50:45.126-7:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>23</Length>
        <X>152.3625 152.775 152.7375 152.5875 152.475 151.425 150.15 148.95 148.4625 148.35 148.0875
                148.0875 148.3125 148.8 149.5125 150.9 151.8 153.3375 154.0125 154.125 154.125 154.0875
                154.0875</X>
        <Y>60 59.8125 59.7 59.3625 59.25 59.025 59.2875 60.0375 60.7875 61.0125 62.1 63.1125 63.975
                64.91251 65.85001 66.15 66.1125 65.55 65.0625 64.875 64.8 64.6125 64.5</Y>
        <TimeStamp>2009-06-15T18:50:48.663-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>12</Length>
        <X>146.85 146.8875 146.8875 146.925 146.925 146.9625 147.5625 149.25 152.3625 152.55 152.5125
                152.4</X>
        <Y>73.425 73.5 73.575 73.6125 73.6125 73.6125 72.2625 67.8 71.775 72.03751 72 71.8875</Y>
        <TimeStamp>2009-06-15T18:50:49.981-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>23</Length>
        <X>148.3125 148.65 149.25 149.2875 149.175 149.4 149.85 150.9 152.0625 152.85 153.5625
                154.0875 154.1625 153.9 152.9625 152.2875 151.65 151.2375 149.25 148.9875 148.95
                148.9875 149.475</X>
        <Y>50.925 51.75 56.25 57.4875 58.0875 55.275 54.1875 53.1 52.8 53.0625 53.7375 54.8625 55.7625
                56.8125 57.8625 58.2375 58.4625 58.4625 58.0875 57.825 57.5625 57.45 57.0375</Y>
        <TimeStamp>2009-06-15T18:50:52.241-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>32</Length>
        <X>157.8375 157.95 157.875 157.6125 157.3125 156.6 155.8125 154.875 154.6125 154.8375 155.625
                156.225 156.75 157.6125 157.9875 158.2875 158.0625 157.9875 158.5125 158.5125 158.3625
                157.9875 156.375 155.55 155.4 155.5125 156.0375 157.875 158.775 158.7375 158.6625
                158.625</X>
        <Y>53.85 53.4 53.2875 52.7625 52.5 52.425 52.7625 53.9625 55.35 56.2125 56.925 57.0375 56.925
                56.2875 55.725 54.375 53.175 57 61.0125 62.2125 62.7 62.8125 62.6625 62.25 61.725
                60.4125 58.8375 56.175 55.2 55.2 55.2 55.2</Y>
        <TimeStamp>2009-06-15T18:50:53.831-07:00</TimeStamp>
        </XMLStroke>
        </Strokes>
```


Figure 4.31: XML file snippet with color to distinguish the different strokes. Blue for “a”,green for “c”, red for “^ ”,yellow for “b” and orange for “g”

Figure 4.32: The strokes after the incorporation of the correction.

Deletion of space

The context of the delete space is two characters between which a space is present. The deletion operation is done by decreasing the X coordinate of the second character by the space between the two characters. The rest X coordinate of the rest of the text line is also decreased the same amount. This space deletion is shown in Figure 4.36.

Movement

The move correction symbol has some characters to move to some other destination indicated by the end of the line as explained in the context extraction section. The characters to be moved are inserted to the position indicated by the end of the line stroke. This shift is done by changing the x-y coordinates. The

```
<Strokes>
        <XMLStroke>
        <Length>33</Length>
        <X>142.275 142.4625 142.4625 142.5 142.5 142.5 142.5 142.5 142.425 141.6 140.925 139.7625
                138.45 137.0625 136.65 136.3875 136.3125 136.725 137.2875 138.3 139.425 141.225 142.65
                142.95 142.9875 142.2375 142.4625 143.1 143.4 144.2625 145.0875 146.2125 146.0625</X>
        <Y>60.75 60.675 60.7125 60.7125 60.7125 60.7125 60.675 60.6375 60.525 59.7375 59.55 59.5875
                60.1875 61.2375 61.8 62.5125 63.7125 65.175 66.1875 66.8625 66.9375 66.2625 64.8375
                63.9375 62.2125 60.2625 63 64.575 65.0625 66.1125 66.525 66.75 66.75</Y>
        <TimeStamp>2009-06-15T18:50:45.126-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>23</Length>
        <X>160.08750610351564 160.50000610351563 160.46250610351564 160.31250610351563
                160.20000610351562 159.15000610351564 157.87500610351563 156.6750061035156
                156.18750610351563 156.07500610351562 155.81250610351563 155.81250610351563
                156.03750610351562 156.52500610351564 157.23750610351563 158.62500610351563
                159.52500610351564 161.06250610351563 161.73750610351563 161.85000610351562
                161.85000610351562 161.81250610351563 161.81250610351563</X>
        <Y>60 59.8125 59.7 59.3625 59.25 59.025 59.2875 60.0375 60.7875 61.0125 62.1 63.1125 63.975
                64.91251 65.85001 66.15 66.1125 65.55 65.0625 64.875 64.8 64.6125 64.5</Y>
        <TimeStamp>2009-06-15T18:50:48.663-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>23</Length>
        <X>148.08749389648438 148.42499389648438 149.02499389648438 149.06249389648437
                148.94999389648444 149.17499389648438 149.62499389648437 150.67499389648438
                151.83749389648438 152.62499389648437 153.33749389648438 153.86249389648438
                153.93749389648437 153.67499389648438 152.73749389648438 152.06249389648437
                151.42499389648438 151.01249389648844 149.02499389648438 148.76249389648844
                148.72499389648436 148.76249389648844 149.24999389648437</X>
        <Y>59.47500305175781 60.30000305175781 64.80000305175781 66.03750305175781
                66.63750305175782 63.82500305175781 62.73750305175781 61.650003051757814
                61.35000305175781 61.61250305175781 62.28750305175781 63.41250305175781
                64.31250305175782 65.36250305175781 66.41250305175781 66.78750305175781
                67.01250305175782 67.01250305175782 66.63750305175782 66.37500305175782
                66.11250305175781 66.00000305175782 65.58750305175781</Y>
        <TimeStamp>2009-06-15T18:50:52.241-07:00</TimeStamp>
        </XMLStroke>
        <XMLStroke>
        <Length>32</Length>
        <X>165.56250610351563 165.67500610351562 165.60000610351562 165.33750610351564
                165.03750610351562 164.32500610351562 163.53750610351562 162.60000610351562
                162.33750610351564 162.56250610351563 163.35000610351562 163.95000610351562
                164.47500610351562 165.33750610351564 165.71250610351564 166.01250610351562
                165.78750610351562 165.71250610351564 166.23750610351562 166.2375061035156
                166.08750610351564 165.71250610351564 164.10000610351562 163.27500610351564
                163.12500610351563 163.23750610351562 163.76250610351562 165.60000610351562
                166.50000610351563 166.46250610351564 166.38750610351562 166.35000610351562</X>
        <Y>53.85 53.4 53.2875 52.7625 52.5 52.425 52.7625 53.9625 55.35 56.2125 56.925 57.0375 56.925
                56.2875 55.725 54.375 53.175 57 61.0125 62.2125 62.7 62.8125 62.6625 62.25 61.725
                60.4125 58.8375 56.175 55.2 55.2 55.2 55.2</Y>
        <TimeStamp>2009-06-15T18:50:53.831-07:00</TimeStamp>
        </XMLStroke>
</Strokes>
```


Figure 4.33: XML file snippet with color to distinguish the different strokes. Blue for “a”,green for “c”,yellow for “b” and orange for “g”

(a) (b)

Figure 4.34: (a)Deletion (b) Replacement

insertion is done in a similar manner as the insertion correction symbol. After the insertion, the text line is rearranged so that the gap created after moving the characters is changed back to the previous gap that was present before moving. This moving action is illustrated in Figure 4.37.

4.6 On-line and Off-line Mode

The system explained in this section was focused on the on-line operation of the digital pen. The distinction between the on-line and off-line mode is important in digital pen correction as mentioned in the introduction, chapter 1. The information which is sent to the server by the digital pen in the two modes is different which makes the two modes different. In the on-line mode the user

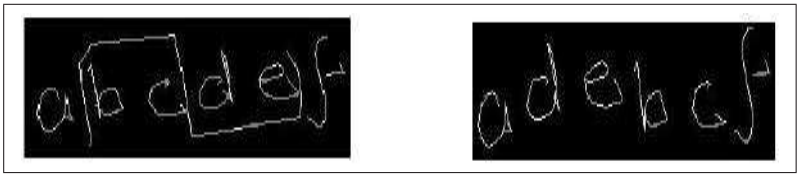


Figure 4.35: Substitution.

Figure 4.36: Deletion of space.

Figure 4.37: Movement.

starts writing and edits the text and the pen generates a single file which is sent to the server. The off-line mode is a bit complicated because after additional writing and editing on some text originally written in the on-line mode, the pen produces a separate single file which is sent to the server as seen in Figure 1.2. This additional information of the new file has to be augmented to the original file from the on-line mode for the correct image to be formed. The augmenting process is simple as the information generated in the off-line mode is extracted and is placed at the end of the information in the file generated in the on-line mode. After augmenting the two files the recognition process is the same as in the on-line mode explained in this section. To implement the process of the off-line mode, the digital paper is required to have a unique pattern so that each paper is identified with this pattern. This unique pattern is usually maintained and once this pattern exists, the file that is sent to the server can be arranged according to the paper easily by checking the time of generation of the files sent to the server.

Chapter 5

Experiment and Results

Two experiments were carried out for our work. One which calculated the accuracy of the three different methods used for the recognition process and the second experiment was the independence chi-square test to check if the errors generated by these three methods were independent of them.

Data for the experiments were collected by manually writing the symbols on the digital paper using the Anoto digital pen as existing database of the symbols does not exist. The digital paper was created by printing the tiny dot patterns on a normal paper.

In the first experiment to estimate the accuracy of the three methods, two different data sets were collected: one for estimating the parameters for the fuzzy functions and the other to test the recognition systems that use the fuzzy functions. For the estimation of parameters for the fuzzy functions, 130 samples were collected with 10 samples per symbol. The features explained in the chapter of feature extraction, Chapter 4.3.2, was extracted from these symbols. The value of the features were then manually analyzed to estimate the parameters for the fuzzy functions in such a way that the trapezoidal shape of the functions overlap each other properly as fuzzy functions usually overlaps. This training data is also used to train the fuzzy network. The network was trained in 5 iterations with zero errors. The testing data contains 650 samples from 5 different users with 10 samples per symbol. Each user was instructed to write 10 samples of a symbol at a time in order to group the samples of a single symbol in one file which the digital pen generates. In such a way we have 13 files corresponding to each symbol with the XML structure described in chapter 4.1, Input, where the x and y node corresponds to the x-y coordinate axis of the symbols.

For testing each method used for the recognition, the files made for testing was selected one at a time and the accuracy for each symbol was calculated for each method. To calculate the accuracy, firstly, the features were extracted from

each sample of the symbol present in a file. These feature values are then used by the fuzzy and crisp functions to assign values to the sets that are used by the input for the recognition modules mentioned in the proposed system, chapter 4. Once the input for the methods is generated, it is given to the recognition module and a classification is made by each recognition module. The accuracy of correct classification is calculated by comparing the classification given by the module and the symbol dedicated to the file from which the sample was taken. The accuracy of the correct classification of the symbol given by each method is shown in Table 5.1.

Method	Accuracy
FNN	94.5%
Decision Tree	93.6%
Min-Max	93.6%

Table 5.1: Accuracy of the 3 methods

From the Table 5.1 we can see that the decision and min-max methods are equivalent in terms of their recognition ability. The benefit of the min-max algorithm is not seen because the features were specific to the symbols hence the decisions at the nodes in the decision tree were representative of all symbols present under it (see Figure 4.15). This shows that the features selected were good because there were no features that were ambiguous for classification. The FNN is slightly better than these two. This is the fact that the FNN was trained and it could make an intelligent decision with the help of weights that were estimated in the training.

Most errors occurred while writing the insert character symbol. Analyzing the symbols the users wrote, it was seen that the user were not able to follow the constraints. This shows that the concept of being “longer” or “shorter” is difficult to express in writing (see in Figure 5.1). One of the main reasons is because the user was not trained.

Figure 5.1: User intended to make the left side longer but failed.

Some errors were also induced by the digital pen. The digital paper which was printed prevented the digital pen to capture some of the stroke information causing the error in the recognition process.

Table 5.2 shows the number of misclassification of the symbol by each recognition method. The misclassification of symbols for the FNN and the other two methods are different. This difference in the errors could mean that FNN could

better classify some symbols better than the other methods. To check if the errors depend on the methods we carried out the second experiment with a larger set of samples as the samples collected for the first experiment was not enough to check the dependence.

Table 5.2: Error for each symbol in FNN, decision tree, and min-max methods.

Method/Symbols	M	ILU	IRU	IU	ILD	IRD	ID	DS	D	S	IS	SV	ISV	Error
FNN	2	1	5	6	6	5	7	3	1	0	0	0	0	36
Decision	1	1	5	9	6	7	9	1	3	0	0	0	0	42
Min-max	1	1	5	9	6	7	9	1	3	0	0	0	0	42

Here,

M = Move	IS = insert space
ILU = insert left up	SV = substitute variation
IRU = insert right up	ISV = insert space variation
IU = insert up	DS = delete space
ILD= insert left down	D = delete
IRD = insert right down	S = substitute
ID = insert down	

In the second experiment to check the independence between errors and the methods, the same recognition system used in the first experiment were used along with the same parameters of the fuzzy and crisp functions determined in the first experiment and a separate set of data were collected for testing. This testing data was collected from a single user with 500 samples per symbol with a total of 5500 samples. The collection of samples and the calculation of the misclassification was done in the same manner of the first experiment. The contingency table for the error produced by each method is given in the Table 5.3.

Table 5.3: Error for each symbol in FNN, decision tree, and min-max methods.

Method/Symbols	M	ILU	IRU	IU	ILD	IRD	ID	DS	D	S	IS	Total
FNN	9	9	3	73	21	19	80	13	18	24	14	283
Decision/MinMax	9	24	14	14	7	85	18	7	80	25	19	302
Total	18	33	17	87	28	104	98	20	98	49	33	585

Here,

M = Move	IS = insert space
ILU = insert left up	ID = insert down
IRU = insert right up	S = substitute
IU = insert up	DS = delete space
ILD= insert left down	D = delete
IRD = insert right down	

From the table we can see that different methods give different distribution of errors. One method is better to classify certain symbols and the other methods for some other symbols. To determine if this relation really exist we used the chi-square test for independence. Using the table we carried out a chi-square test for the independence with a null hypothesis stating that the error produced by the methods are independent of the methods. This test rejected the null hypothesis with a confidence of more than 95% indicating that the errors produced by the methods are dependent on the methods. This shows that using different methods with the same basis of classification, the fuzzy rules, the methods gives different errors for different symbols which suggest that some methods are good for the classification of certain symbols and the other method for the rest. With this information we know that the accuracy can also be increased by merging the methods together.

Chapter 6

Conclusions

In this work, we present a new system that automatically incorporates the vocabulary corrections that the user makes while writing with a digital pen. For the vocabulary correction we propose 13 correction symbols. Our system differs from existing editing systems by the fact that our system can handle information that consists of both text line strokes and correction strokes. In existing editing systems, the information of the text line strokes and the correction strokes are already separated because of the nature of the editing process of digital documents.

To achieve this ability to incorporate the corrections made by the user, our system separates the text line strokes and the candidate correction strokes. This separation is done by analyzing the spatial position of the strokes. The candidate correction strokes are then sent to the recognizer to recognize the symbol. The complexity and processing of the recognition is simple and fast due to the elimination of the preprocessing part. The recognition of symbols is done using three simple algorithms: FNN, decision tree, and min-max algorithm, which gives an accuracy of 94.5%, 93.6%, and 93.6% respectively. These algorithms use fuzzy sets which are created by fuzzy functions which uses 10 features of the stroke. As the number of symbol increases, the difficulty of selecting the best features for the classification increases. Among the three methods used, fuzzy neural network has the highest accuracy, but does not differ much with the other two methods. For simplicity and for having an option of being able to get a ranking of recognized symbols, min-max algorithm would be a better choice to implement the correction system. With the ranking option, the system can be extended to an interactive correction system to select the lower ranked symbols. After the symbol is recognized, its context is extracted and the correction is incorporated in the XML file. An example of the incorporation process is given in the Appendix A. To minimize the error of this approach we propose that the user has to be trained

to correctly follow the constraints of symbols. We also determined that different method are good for recognizing certain symbols. This information leads us to merge the methods in such a way that we could recognize certain symbols with certain methods which will also increase the accuracy of the system.

There are still some open issues to consider for vocabulary correction. One of them is to handle corrections over corrections, which our system cannot handle. The issue about the skewed texts is also still not completely resolved. The correction symbols are sensitive to its rotational orientation so, if the text is skewed the correction symbols may not be correctly recognized. Another issue is the natural habit of rewriting of segments in a stroke, which our system assumes does not occur. Normally, some segments of a stroke are written twice for example, in the caret sign for upward insertion " ^ ", we have a stroke that has the first segment which goes from bottom to top and another segment which goes from top to bottom and some have habits of rewriting the first segment from top to bottom and bottom to top. These issues are not handled by our system and could be handled in future work.

Bibliography

[1] http://actimage.de. [cited at p. 2]

[2] http://en.wikipedia.org/wiki/decision_tree. [cited at p. 23]

[3] http://en.wikipedia.org/wiki/digital_paper. [cited at p. 2]

[4] http://en.wikipedia.org/wiki/digital_pen. [cited at p. 1]

[5] http://en.wikipedia.org/wiki/fuzzy_logic. [cited at p. 18]

[6] http://en.wikipedia.org/wiki/minimax. [cited at p. 24]

[7] http://en.wikipedia.org/wiki/polynomial_interpolation. [cited at p. 10]

[8] http://www.erasme.org/img/stylo_numerique.pdf. [cited at p. 2]

[9] H. S. M. Beigi. An overview of handwriting recognition. In *Proceedings of the 1st Annual conf. On tech. Advancements in developing countries*, 1993. [cited at p. 10]

[10] Marotte Jean-Baptiste Beugnot Julien and Vaziri Vahid. Vocabulaire de correction de son criture. Technical report, University Nancy2, 2008. [cited at p. 6]

[11] H. Bunke, R. Gonin, and D. Moeri. A tool for versatile and user-friendly document correction. In *Proceedings of the 4th international conference on document analysis and recognition*. ICDAR, 1997. [cited at p. 9]

[12] M. Caudill and Charles Butler. *Understanding Neural Networks vol.* vol. 1 and 2 edition, 1992. [cited at p. 25]

[13] T.Wakahara C.C. Tappert, C.Y. Suen. The state of the art in on-line handwriting recognition. In *IEEE transactions on pattern analysis and machine intelligence*, volume vol. 12, 1990. [cited at p. 3, 4]

[14] B. B. Chaudhuri. *Document Processing: Major Directions and recent advances.* Springer, 2007. [cited at p. 3]

[15] R. Genoe, J.A. Fitzgerald, and Tahar Kechad. An online fuzzy apprach to the structural analysis of handwritten mathematical expressions. *IEEE int. conf. On fuzzy systems*, 2006. [cited at p. 11]

[16] F. Guimbretiere. Paper augmented digital documents. *Human-Computer Interaction Lab.University of Maryland.* [cited at p. 4, 9]

[17] K. Gurney. *Introduction to neural networks.* CRC Press, 1997. [cited at p. 25, 26]

[18] http://kmil.trios.cz/korekce1.htm. [cited at p. 5]

[19] http://neuroph.sourceforge.net/. [cited at p. 13]

[20] http://www.colorado.edu/Publications/styleguide/symbols.html. [cited at p. 4, 5]

[21] http://www.e write.de/mg/downloads/data/pdf/ewrite/korrekturzeichen.pdf. [cited at p. 6]

[22] http://www.typo.cz/_typo/images/korektorske znacky.pdf. [cited at p. 6]

[23] J.Andre and H. Richy. Paper-less editing and proofreading of electronic documents. *EuroTEX proceedings*, 1999. [cited at p. 4, 9]

[24] D. Levein K. M. Conroy and F. Guimbretiere. Proofrite : A paper- augmented word processor. *Human-Computer Interaction Lab.University of Maryland.* [cited at p. 4]

[25] A. Kankaanpaa. Fids : A flat-panel interactive display system. *IEEE Computer Graphics and Applications*, vol. 8, 1988. [cited at p. 9, 10]

[26] M. Liwicki and H.Bunke. Hmm-based on-line recognition of handwritten whiteboard notes. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, 2006. [cited at p. 10, 16]

[27] N. Kharma M. Cheriet, C. Liu, and C.Y.Suen. *Character recognition systems: A guide for students and practitioners.* John Wiley and Sons, 2007. [cited at p. 3]

[28] E. Indermuhle M. Liwicki and H. Bunke. Online handwritten text line detection using dynamic programing. In *Document Analysis and recognition.* ICDAR, 2007. [cited at p. 15]

[29] S. Bagheri Shouraki M. Soleymani Baghshah and S. Kasaei. A novel fuzzy classifier using fuzzy lvq to recognize online persian handwriting. In *2nd Intl. Conf. on Info. and Com. Tech.: from Theory to Applications.* ICTTA, 2006. [cited at p. 10]

[30] A. Malaviya and P. Malaviya. Object recognition using fuzzy set theoretic techniques. *SPIE Proceedings*, Vol. 1962, 1993. [cited at p. 11]

[31] A. Malaviya and L. Peters. Feature description of handwriting patterns. *Science Direct: Pattern Recognition*, Vol. 30, 1996. [cited at p. 11]

[32] A. Malaviya, L. Peters, and R. Camposano. A fuzzy online handwriting recognition system: Fohres. *2nd int'nl conference on fuzzy theory and technology*, 1993. [cited at p. 11]

[33] Thomas Mitchell. *Machine Learning.* 1st edition edition, 1997. [cited at p. 23]

[34] Witold pedrycz and Fernando. *Gomide An introduction to fuzzy sets: analysis and design.* MIT Press, 1998. [cited at p. 18]

[35] A. Pikaz and I. Dinstein. An algorithm for polygonal approximation based on iterative point elimination. *Pattern Recognition Letters*, vol. 16, 1995. [cited at p. 11]

[36] R. Plamondon and S.N.Srihari. Online and offline hadnwriting recogntion: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2000. [cited at p. 16]

[37] J. Rhyne. Dialogue management for gestural interfaces. *ACM SIGGRAPH computer graphics*, vol. 12, 1997. [cited at p. 9]

[38] H. Richy and G. Lorette. On-line correction of web pages. In *Proceedings of the 5th Conference on Document Analysis and Recognition*. ICDAR, 1999. [cited at p. 4, 9]

[39] D. Rubine. Specifying gestures by example. *ACM SIGGRAPH computer graphics*, vol. 25, 1991. [cited at p. 9]

[40] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002. [cited at p. 24]

[41] T.Wang, Y.Yan Z.Li, and H.Chen. A survey of fuzzy decision tree classifier methodology. *Proc. Of the 2nd int. conf. On fuzzy information and engineering (ICFIE)*, 2007. [cited at p. 23]

[42] M.T. Manry Z. Wang and J.L. Schiano. Lms learning algorithms: misconceptions and new resutls onconverence. *IEEE trans. On Neural Networks*, Vol. 11, 2000. [cited at p. 25]

[43] L. A. Zadeh. Fuzzy sets, information and control. 1965. [cited at p. 4, 18]

[44] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 2003. [cited at p. 10]