An Overview of Handwriting Recognition

Homayoon S.M. Beigi
T.J. Watson Research Center
International Business Machines
P.O. Box 704
Yorktown Heights, New York 10598
EMail: beigi@watson.ibm.com

Keywords: Handwriting, Recognition, OCR, Signature Verification

Abstract

This is an overview of the most recent published approaches to solving the handwriting recognition problem. This paper is aimed at clarifying the role of handwriting recognition in accordance with today's maturing technologies. It tries to list and clarify the components that build handwriting recognition and related technologies such as OCR (Optical Character Recognition) and Signature Verification. This paper could also be regarded as a survey of handwriting recognition and related topics with a rich list of references for the interested reader. Levels of practicality of use of this technology for different languages and cultures is also discussed.

1 Introduction

As portable computers become more personal and are made smaller, they reach some physical limitations for having a keyboard. For pocket size computers (PDA's) one cannot use an efficient keyboard. For these computers alternative ways of man-machine communication are necessary. A most efficient way of solving this problem is to use communication skills of man which have developed for thousands of years namely, speech [1] and handwriting. There are certain merits and drawbacks to both of these techniques thus comprising the reason why we still communicate using both methods. One of the most obvious reasons that handwriting recognition capabilities are important for future personal systems is the fact that in crowded rooms or public places one might not wish to speak to his computer due to the confidentiality or personal nature of the data. Another reason is that it might be annoying to us if someone sitting next to us in the train or airplane keeps speaking to his machine. Another reason for the practicality of a system which would accept hand-input is that with today's technology it is possible to have handwriting recognition in very small hand-held computers, however speech systems could not yet be made so small as to fit in a standalone hand-held machine. One of the most important merits of speech systems is however apparently the speed of data entry. It is much easier to dictate something than to write it.

In regard to desktop computers, neither handwriting nor speech would necessarily have to replace the keyboard. They could however be very useful complementary devices to the keyboard. In pen-computers, the pen is used to do the job of a mouse in addition to its specific task of handwriting input. In addition, use of a pen instead of the mouse avoids having to learn fancy hand-eye coordination since in most pen-computers, the user writes directly on the display device.

Many data types are easier to input by pen than by keyboard. Examples of these data are mathematical equations, music notes, free-hand drawing, etc. Also, filling a form is easier with a pen than with a keyboard since one could directly go to the appropriate field and make his entry.

Gestures form another very important class of pen-traces that we use in our every day life and it would be possible to use them with pen-computers. As an example, editing something on paper, we make all types of markings which have found standard meanings. Pigtails for deleting a character, crossing a paragraph for deleting it and a circle followed by an arrow for moving a paragraph from one location of the paper to another are examples of these markings. Gesture recognition allows for all these features to be used with computers when creating or editing a document. Also, within a document, one could change from text to graphics without having to change his input device or having to resort to complicated programs. [2]

An even more important merit of pen-computers is that anyone who does not know how to type could still use the computer. Typing illiteracy might not be very common in many western countries like the United States, however, it poses a big problem in countries like Iran or Arabic countries in which almost only secretaries know how to type in their native alphabet. This limits the amount of computer usage in those countries. However, one deciding factor that gives the second set of countries an easier choice in deciding what to use for their input device is that computers are not very widespread in these countries. People of these countries could therefore decide to use pen-computers from the early stages of computerizing their institutions.

Most of the above discussion assumes that the input to the computer is an on-line trace of the points a pen-tip will traverse as the writing takes place. Another method of recognizing handwritten or even typewritten text is to scan the image of the written or typed document and to have the computer recognize the content of it. This, known as OCR (Optical Character Recognition), is a good method of extracting information from our archives which have been generated over hundreds, even thousands of years. Another important application that has kept many researcher working on this problem is the postal sorting problem. Every day there are millions of mail pieces being delivered by different postal services around the world. In sorting these letters, most of the time a human reader is used. Many countries have recently adopted automatic sorting techniques that will recognize the addresses on the mail pieces and sort them accordingly.

Another important technology which is going to be briefly touched here is signature verification. Every year, many millions of dollars are lost to fraud which could be prevented by more strict signature verification strategy. Many store clerks do not usually check the signature of a customer against that on his credit card. Not being signature verification experts, it is apparent that these clerks' verification would be superficial even if it were conducted as required. Signature verification if done on-line, could use the same pen-computers as discussed earlier to do a much better analysis of the signature than any human specialist will ever be able to do. Computer driven signature verification systems have much more information available to them, such as the local velocity of the pen-tip at different portions of the signature.

This paper will provide some information about on-line handwriting recognition, OCR and signature verification.

2 On-line Handwriting Recognition

In an on-line handwriting recognition system, the motion of the tip of the stylus (pen) is sampled at equal time intervals using a digitizer tablet and it is passed to a computer which runs the handwriting

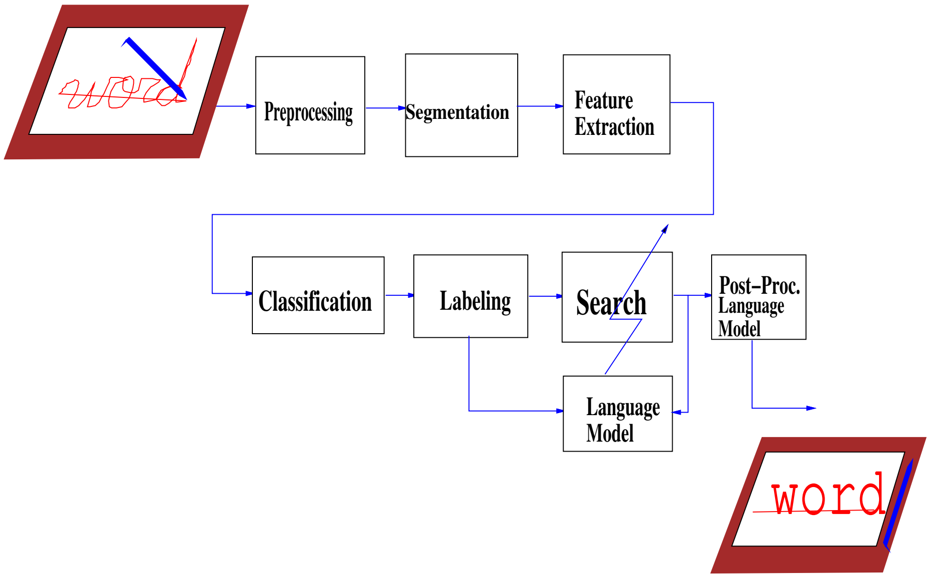

Figure 1: Generic Handwriting Recognition Process

recognition algorithm. In most systems, the data signal undergoes some filtration process. Then the signal is normalized to a standard size and its slant and slope is corrected. After normalization, the writing is usually segmented into basic units and each segment is classified and labeled. Using a search algorithm in the context of a language model, the most likely path is then returned to the user as the intended string (see Figure 1).

2.1 Digitizers

Digitizer technology has always been one of the most important bottle-necks in the development of on-line handwriting recognizers. This technology started off in the late 1950's [3]. The first digitizer tablets were bulky and opaque. Their sampling rates were very low (less than 70 Hz) and they generated nonlinear signals depending on the position of the stylus on the tablet. As digitizer technology matured, by the late 1980's and early 1990's, less expensive digitizers were developed which were very small and practical with reduced nonlinearities. These new digitizer made better handwriting recognition accuracies possible. Many different strategies have been used for building digitizer tablets. Some of these methods are inductive, capacitative, piezoresistive, conductive, and electromagnetic stylus tablet interactions.

With the enhancement of Liquid Crystal Display (LCD) technology, digitizer tablets were placed under the LCD displays such that a transparent look could be given to the writing on these tablets. With these new digitizers, the trace of the stylus could instantly be displayed on the LCD display acting as an electronic ink. These tablets were much easier to get used to for the untrained user. There were still some problems which needed a solution such as sufficient backlighting, parallax due to the thickness of the LCD's, tablet nonlinearities, excessive weight, slippery glass surfaces, pen design and ease of handling, etc. Most of these problems have received some very advanced and acceptable considerations and solutions. Digitizers are now transparent. They have minimal parallax and reduced nonlinearities. some have great color displays. Manufacturers have induced friction between the tip of the stylus and the glass over the LCD to simulate a pen-paper like feel. Most important of all, these digitizers are now much less expensive and they can produce sampling rates of over 200 Hz.

Most experiments reveal that sampling frequencies over 100 Hz are well sufficient for obtaining good recognition results. For a good survey of digitizer technology please see [4].

2.2 Preprocessing

Most digitizer tablets have built-in low-pass filters in hardware form which take away the jagged nature of the handwriting signal. These filters in some cases generate more problem than they solve. For example, the smoothing of the data at the hardware level could sometimes smooth away natural cusps and corners which are very important in recognizing certain characters such as v's and s's. It is sometimes more desirable to do minimal filtering at the hardware level and to leave most of the filtering to the discretion of the recognition algorithm developer.

Once some basic filtering is done, it is usually desirable to magnify the writing to a standard height such that the recognition scheme could become size independent. [5] To perform such normalization, the base-line and mid-line should be estimated (see Figure 2).

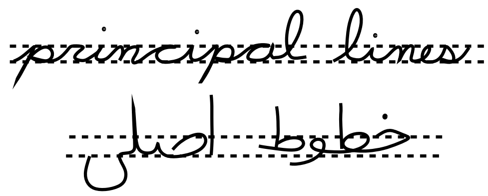

Figure 2: Principal lines of a word

The area surrounded by the the base-line and the mid-line is the only part of any word which is always non-empty. This makes this area the most reliable portion of the data for usage in size normalization. Once accurate estimates of the base-line and the mid-line are given, a magnification factor could be computed from the ratio of the nominal mid-portion size and that of the input. The entire input data may then be magnified using the obtained magnification factor.

Other possible normalizations are slant and slope correction. In slant correction, usually some mean dominant vertically oriented slope is computed. The slope of the data is then offseted using the difference between a slope of the vertical axis and the computed slope. This correction is usually done through shearing since for small deformations shearing is a good approximation for rotation.

Slope correction is usually an iterative process which uses both of the above normalizations to estimate and re-estimate the slope of the base-line and then the data is slope-corrected by shearing it along the vertical axis such that the base-line becomes horizontal.

2.3 Segmentation

Hand input is classified into five different types. Figure 3 shows samples of these types of input. In case of boxed-discrete input, basically the writer is segmenting his writing into separate characters. This is probably the simplest form of writing to be recognized. In the second type, the writer once again aids the recognizer in segmenting the writing into individual characters. In this case, the problem of segmenting the data into separate characters is solved by finding those gaps between successive chunks of data in the horizontal direction which are greater than a predefined or statistically obtained threshold. [6]

In run-on writing, the problem of segmenting the word into characters becomes nontrivial. In this case, the characters could even overlap such that gap information is no longer sufficient for character segmentation. The only restriction which is imposed on the method of writing run-on is that the pen should be lifted from the surface of the digitizer after each individual character is inputted. One solution to this problem is to treat each stroke of the writing as the smallest unit of the word and conduct a search through reasonable combinations of the stroke labels which could create legal words. [6, 7]

The next type of writing is pure cursive which has even less restrictions imposed on its methodology. For pure cursive, the only two restrictions which are imposed are that there is a pen lift at the end of each word and that all characters are connected to their adjacent character.

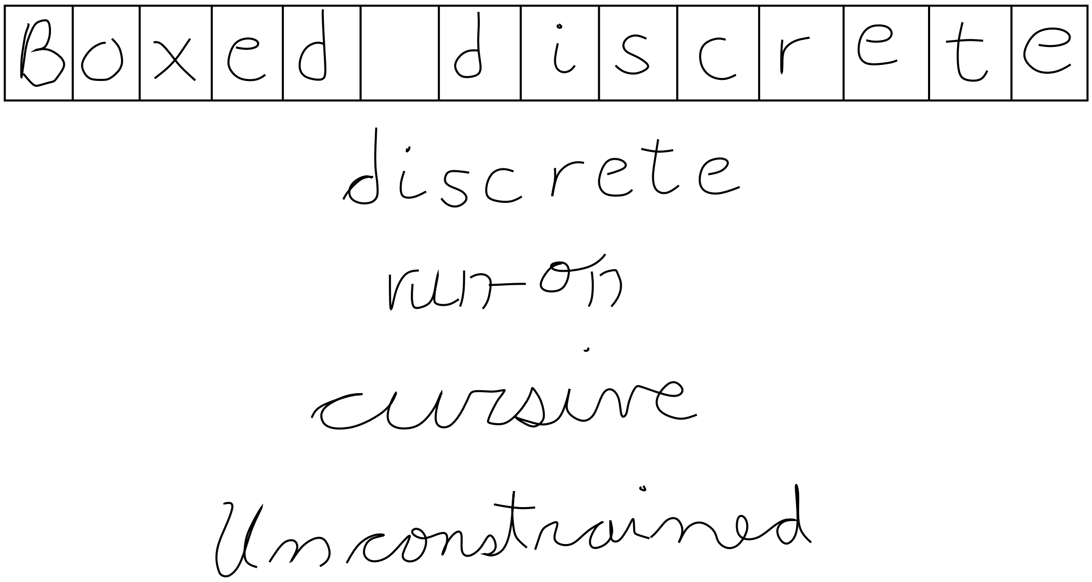

Figure 3: Samples of Different Types of Handwriting

Most people, however, write in a combination of cursive and run-on. This writing style has no limitations imposed other than in some cases there should be a pen-lift after each word. This is the ultimate challenge in handwriting recognition which has attracted lots of attention recently. [8] Companies such as IBM, Paragraph, Lexicus, CIC and so on have created cursive and unconstrained recognition systems which are either available or will soon be available to the general public. For recognition of cursive and unconstrained writings, segments are often defined to be a subset of a character and they are generated based on some criteria set by the recognition algorithm. Some systems segment the input based on predefined sub-characters such as those given in [9]. Many others assume that handwriting is generated by two coupled oscillators by nature [10] and segment the data at points of extreme velocities – usually at minimum y-velocity. These segments (usually sub-characters) are then sent to the core of the recognition engine for labeling and hypothesis generation. Many other types of segmentation have been used such as trying to find optimal points of character breaks or ligature spotting. [11]

2.4 Feature Extraction and Shape Classification

Once the writing is segmented into smaller units, these units are sent to a module which extracts features in the data, essential to the employed shape classification algorithm. These segments are ei-

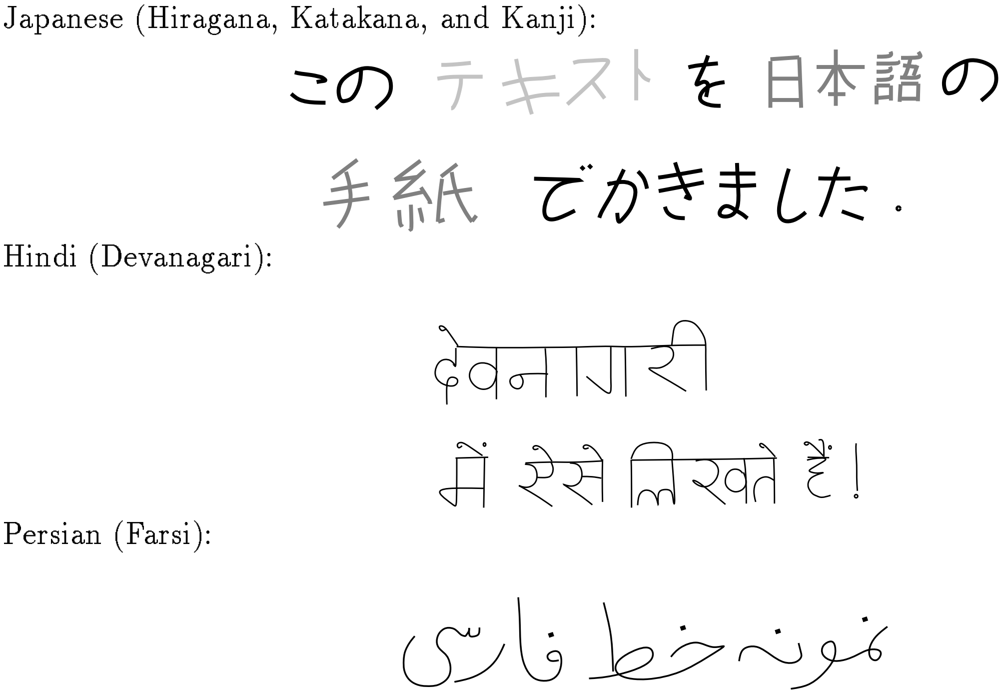

Figure 4: Some Handwriting Samples for Different Languages

ther strokes, or sub-strokes and they usually carry information such as the $< x, y >$ coordinates of the points and their time-stamps of the points which carry the speed and order information of these points.

Some recognition schemes throw away the speed information of the data and keep only the order information of the points. In this sense, these recognizers use mainly spatial features. [7, 12, 9, 13, 14] On the other hand there exist systems which dominantly use the temporal information in the data. [15, 16]

Some spatial features which are widely used by researchers are differences between adjacent coordinates (Δx and Δy), slope of the tangent line at each point (this could be given as an angle), and sometimes the curvature value at each point. The slope and curvature information could be evaluated by looking at adjacent points. Some systems also use the coordinates of the points with respect to the computed base-line or some other reference point. This information could be used for evaluating the size of the writing such that with some simple comparison, the difference between upper and lower case characters could be recognized. A class of recognizers use the coordinate information to build features such as population of points in specific predefined zones. This zonal feature extraction is mainly used in off-line systems. [17] In some systems, stroke direction and zonal information are usually noted for building the features. [18]

In the on-line recognition of handwriting, many systems make use of the dynamic information that is available to them. These systems usually approximate the handwriting by the output of a dynamic model.[10, 15, 16] The parameters of the model such as frequency, phase and amplitude are then evaluated by considering the dynamic information available in the segments of data. These parameters are then used as compressed features to be passed to the classifier for labeling.

2.5 Labeling

Once the features are generated for corresponding segments, these segments are usually labeled using a variety of techniques. Stroke-based systems usually label each stroke with lists of hypothesized characters that the stroke may represent. In these systems, a stroke is defined as a subset of a character. Each character on the hypothesis list associated with each stroke has a probability or likelihood value associated with it. Similarly, for other types of sub-character segments created by the segmenter, an equivalent list of character hypotheses with associated likelihoods could be generated. This hypothesis list generation is done using some classification scheme. Among these different classification and labeling techniques, Template matching, Statistical and Nonlinear Classifiers are described. For a good survey of different sub-character classifiers see [4, 11], [19].

2.5.1 Template Matching Models

At the training stage, template matching systems create certain templates of their elementary handwriting segments and represent them by the feature parameters used for recognition. At the recognition stage, these systems compute some distance measure from the segment of data to the template segments and for each template, they generate a likelihood value depending on the closeness of the input data to the template. One very popular matching algorithm which has been used widely in the literature is elastic matching. [20] The basic concept behind elastic matching is the computation of the minimum distance measure between a set of points and those of a template which does not necessarily have the same number of points. Stroke based systems [6] match a stroke against a pool of stroke templates. These stroke templates are then labeled such that they carry their character information and corresponding position. For instance, a straight vertical line could be labeled as the first stroke of a three stroke upper-case H, the second stroke of a two stroke T, or the third stroke of the same upper-case H, etc. Stroke-matching methods could not be used for recognition of cursive or unconstrained writing since in those cases a character is usually a subset of a stroke. However, there is an exception to this statement. If the template is based on complete words and not characters, then a stroke is always a subset of the word and it is possible to do cursive or unconstrained recognition.

Another recognition system based on template matching relies on the theory that all handwriting is made of some elementary shapes. The segmentation process breaks the writing up into these basic elements and the classifier labels them based on the templates in a fashion similar to that in the above stroke-matching system. These systems could be used for conducting cursive or unconstrained recognition as well as lower complexity writing styles. For examples of these systems see references [9], [14], and [20].

2.5.2 Statistical Models

Statistical models such as hidden Markov models (HMM's) have been used for many years in speech recognition. [1] Recently, these methods have started being applied to handwriting recognition. [12, 18, 21] In these systems, Markov models of different characters are built at the training stage. The number of states and the probabilities associated with different state transitions and outputs are learned from a training data using techniques such as forward-backward (Baum-Welch) maximization. [22, 23] These models could be built such that they would model the duration of staying at a single state by having null and self transitions. Null transitions are used for skipping states while self transitions (self loops) provide means of staying at a certain state for a longer duration.

2.5.3 Neural Network Models

Neural net models have also been used in labeling the segments. In discrete character recognition, some neural net based systems train the weights of template neural nets such that each would generate the supported alphabet with highest likelihood for the data they are trained on. For example, there would be a network which would generate A with highest likelihood compared to all other characters in the set. In decoding, through inputting the data into these template networks, different likelihood values are generated. By using appropriate normalization, these numbers could be used to label the data.

The systems based on the above methodology can only be applied to the recognition of cursive and unconstrained words if templates of each word are made. This is due to the problem of character segmentation that exists in cursive and unconstrained handwriting. To handle unconstrained words, most researchers use Time-Delay Neural Networks (TDNN's) to generate labels for segments of data using a hierarchical network of two basic nets. TDNN's have also been used in speech recognition. [24] A set of identical networks are used to look at adjacent segments of data. The outputs of these adjacent networks are then passed along to another network which has the same number of outputs as the supported alphabet. Each of these outputs is associated with a distinct member of the alphabet and its activity is representative of the likelihood of the data being a part of that character. [25, 26, 27].

2.6 Fast-Match Pruning

For performance reasons it is good practice in any of the above systems to incorporate a fast-match algorithm to prune a number of labels which are generated. The idea behind a good fast-match algorithm is to prune the possibilities with little computational effort. It should also insure that the correct label would not be pruned out. Therefore, the fast-match algorithm should be designed in a very conservative manner. Depending on the efficiency of the fast-match algorithm, the recognition time could be considerably reduced. Usually, these speed considerations are very important in being able to keep up with the speed of the writer in an on-line recognition system.

2.7 Delayed Strokes

In natural handwriting, sometimes, cross-bars in t, dots in i and j, the right to left stroke in x are inputted after writing other letters to their right. These delayed strokes can cause difficulties in on-line handwriting recognition where the temporal sequence of the signals is preserved. In the case of delayed strokes, portions of a letter may be located far apart in time.

Different systems deal with this problem in a different manner, but almost all methods involve first, the realization of the delayed stroke and second, some treatment to take it into account. One practical method is to reorder the segmented data pieces such that the delayed stroke segments are placed in their appropriate positions. [8] Delayed stroke handling is an important feature which any practical recognizer should possess.

2.8 Search

Once lists of hypothesized characters are established for each data segment, a search technique should be employed to find the most likely path through the hypotheses. This path would be the answer string returned from the recognition engine. Many different possibilities exist for conducting this type of a search. It is very important that the employed search algorithm be admissible such that the correct path is not easily lost. [28] Many systems use dynamic programming techniques to find this path. Most dynamic programming methods end up exploring all the possible paths which for on-line handwriting recognition would not be very practical. In fact most systems spend a major part of their time in search.

A useful, admissible search technique that has been used in this field is the A^* search algorithm. [7, 8] This is a left to right search technique which only explores paths of minimum cost (maximum probability). There are still some search errors associated with this technique. Since A^* is a linear search algorithm, it might prefer paths which do well in the first few segments and then get worse nonlinearly. In these cases, it will miss out on the paths that might have low probabilities initially and would pick up later on. This could happen for example if the first couple of characters in the writing are formed poorly and the rest of the word is written more neatly.

Some of these problems could possibly be alleviated by using search algorithms which start out in some neatly formed part of the word and expand to the right and left. These starting regions are called islands of reliability. [29, 30] These techniques, however, have some impracticalities associated with them which are outside the scope of this paper.

2.9 Language Models

2.9.1 Predictive Models

It is possible to reduce the number of hypotheses which are explored by the search process. One way is to take advantage of the statistics available on the likelihood of certain characters following a given string of characters. Given these statistics, impossible paths could be pruned out and the less likely hypotheses could get less priority in their expansion. This could speed up the search process and increase the accuracy of recognition. [31, 32, 7]

Word-level language models have mostly been used in speech recognition systems. [1] These models work in the same manner as the above character model by using statistics of certain words following a string of given words and grammatical restrictions. These ideas have started being used in handwriting recognition. [33] Word-level language models might not be very useful for everyday usage in handwriting recognition. Given the slow nature of writing, most people use handwriting recognition in an ungrammatical manner with lots of abbreviated and broken sentences. It is very difficult to handle these situations if a grammatical restriction is imposed. To solve this problem, [33] uses a corpus of data obtained from electronic mail correspondences at random. The idea behind this choice is that electronic mail is usually very informal and portrays the same style of writing as might be used on a pen-computer.

2.9.2 Template Models

In some cases, the recognizer is expecting a specially syntaxed entry or it is expecting a subset of the normally supported characters. Using this infirmation, many search paths could be pruned out in favor of faster recognition speeds and higher accuracy. An example of such cases is a field which expects an auto license number. Most countries have a set format for their licenses. This format could be used in re-estimating the probabilities of members of a character hypothesis list. Some systems totally prune out illegal characters while other allow some flexibility for exceptions. [34]

2.9.3 Post-Processing

Some efforts have been made to correct a word which is returned from recognizer. In most cases, a dictionary of highly frequent words is used to find the closest string to that returned from the recognizer. Post-processing can increase the recognition accuracy considerably. See references [35], [36] and [37] for such post-processors.

2.10 Vocabulary Size

Almost every recognition system which handles cursive and unconstrained writing, restricts itself to a given set of words at each instance. This helps increase accuracy and speed of recognition. However, once the number of words in an allowable dictionary gets large, keeping up with the writer in an on-line recognition system will become almost impossible. To alleviate this speed problem some holistic pruning algorithms have been developed which would reduce the number of legal words in a dictionary and therefore increase the speed of recognition. [38, 39] These algorithms look at macro features such as descenders, ascenders, loops, etc. This is very similar to what speed readers usually do while reading.

2.11 Training

Almost every recognition system must undergo a training procedure before it is able to operate. This training could be portrayed by establishing some rules of recognition, templates, or statistical and network model training. Some researchers argue that the learning should be done in a generic fashion such that the system should learn to recognize the writing of a wide sample of the society. From that point on, any person using the system should adapt to the machine rather than the machine adapting to his writing. Another group of people argue that we should not change our handwriting style just because we would like the computer to recognize our writing. This group of people argue that the machine should have training capabilities such that if it cannot perform well on a certain word or on a character, the user would be able to train the system. This training could usually be in batch mode or with some systems it could be done on-line.

3 Optical Character Recognition (OCR)

OCR is a very useful tool for transforming printed text or handwriting into text that the computer could retain in ASCII, UNI-CODE, or similar formats. A most attractive feature of this capability is the great compression rates that could be achieved in storing such data. Another reason for the importance of OCR is that it could be used for bringing any document that has been generated before the beginning of the computer era in compatibility with the new generation of computer based data-management. A very important application of OCR is the use of this technology for sorting mail pieces in post offices around the world. Millions of mail pieces are delivered each day and the sorting has conventionally been done by post-office workers. In many countries today, a great portion of this job is being done by OCR systems installed in key postal locations. [40]

An OCR system usually gets an image of the data from a scanner in bit-map form. The data is then segmented into pieces using different algorithms. These different data types could be images, text, etc. [41] Once these different types of data are segmented, the text is segmented further into words or characters and is sent to the recognition engine. The engine performs a skeletonization [42] and a preprocessing similar to those in on-line recognition discussed in the previous section. [41] Different shape classifiers are then employed as in the previous section which extract certain features and create character hypothesis lists. A search algorithm is then used in conjunction with language models to return a resulting string. For detailed information please see [41], [42], [43], [44], [45], and [46].

4 Signature Verification

Signature verification is another related technology which has lots of potentials uses. Every year millions of dollars are lost by credit-card companies to fraud. This is mostly due to the fact that store clerks either do not verify the signature of the user against that on the back of his card or even if they did they would not have the expertise to tell apart a good signature from a forgery. Even experts could be fooled by the signature of a good forger. Signatures, however, have special features that are only duplicable buy their owners. These features are consistencies in local speed, acceleration and pressure in most parts. These features are not apparent from the image of a signature. If the signature is captured using a digitizer tablet some of these features could be captured and trained for.

A few very good signature verification systems have been developed in the middle 1980's. [47] One major problem associated with these systems is that they require very sophisticated digitizers which would return pressure, acceleration and so on. It is desirable to have a signature verification system that would operate on a similar type of data as is sent to a handwriting recognizer, namely, the $< x, y >$ coordinates of the points and their time-stamps. This would enable a wide usage of signature verification in a more practical setting, namely a simple pen-computer. [48] A good strategy is to use the speed information which is available from the time-stamp. The speed could hardly be forged since it relates to the motor-control of the writer and it is not something that could easily be observed by the forger.

5 Other Related Recognition

Other specialized research efforts have been on their way for years. To name a few, research is being done in music note recognition, mathematical formula recognition, hand-drawn flow-chart recognition, etc. [49, 50, 51, 52] With the development of new pen-computers, needs of icon retrieval systems have also become clear. Using these systems, instead of only associating a textual name to a file, an icon could be drawn for it. It is necessary to be able to retrieve any icon by drawing parts of its contents. Reference [53] presents a method for conducting such recognition.

Some research efforts have also been concentrated on the problem of digit recognition. This problem is specially important in applications such as recognition of check amounts. By having a dedicated digit recognition system, the number of errors could be reduced for this delicate recognition process. [54]

A very important feature of pen-computing is the capability to accept and recognize gestures. Gestures are markings such as pig-tails, crosses, etc. which are made to convey a change in the document. In a pen-computing environment, the writer (user) may circle a paragraph and draw a line to a new position in the document. This gesture will move the entire content of the circle to the new location. Also, the user could cross out a paragraph or a portion of a line to delete the crossed data. He could also use a pig-tail to delete a character within a word. These features are possible through effort in gesture recognition. [2]

6 Data

A very important problem in developing recognition systems is usually the insufficiency of data. In the process of creating a recognizers different types of data are used, such as, a text corpus for language model generation, training data, test data, etc. This is specially hard for languages which are used in places where computers are not widely used such as in Iran, Arab countries, China, India, etc. For these languages, it is very time consuming to generate language model or to put together a training data. Some active research is being conducted in the area of standardization of test data, cleaning of large data bases, etc. [55, 56]

7 Pen-Computing and Special Pen Operating Systems

Many pen-based operating systems have been made available to the general public in the past few years. These operating systems are either pen-centric or pen-aware. Pen-centric operating systems such as Pen Point^{TM} have been designed around the existence of a pen as the primary input device. Pen-aware operating systems, on the other hand, are extensions of already existent operating systems with the addition of a pen as just another input device. Windows for Pen^{TM} and OS2 for Pen^{TM} are examples of pen-aware operating systems.

Pen-centric operating systems take full advantage of the capabilities of a pen. However, they have the limitation that all application softwares should be written explicitly for their new architecture. On the other hand, pen-aware systems sacrifice the full capability of a pen for being able to use the thousands of applications already available for them in the market.

8 Conclusion

Handwriting recognition technology, both on-line and off-line, has reached its initial stages of practicality. This is widely due to the advancement in digitizer and portable computing technologies. Much faster processors are available which could also utilize larger amounts of memory. Digitizers are inexpensive and accurate with reduced non-linearities and higher sampling rates. In addition to the hardware technology, more practical recognition algorithms have been developed. Together with the new pen-computers and pen-based operating systems, it is a very good time to take advantage of the capabilities of pen-computers. PDA's have developed to a point that it is very plausible to have a handwriting recognition system on a stand-alone PDA.

8.1 Applications

Handwriting recognition allows more efficient drafting and document generation as well as applications such as form filling and keyboardless interface with a computer. In desktop systems, the pen could be a very important complement to the keyboard for editing, marking, drawing, etc. As the handwriting recognition technology becomes more mature, applications such as longhand note-taking in the class-room are going to be more of a reality. Professionals could write their documents and have them converted to text instantly without having to go through a few iterations of having their secretary type the document for them.

OCR could speed up postal delivery of mail pieces. Check amounts could be recognized automatically and without any human intervention. Signature verification could be done on-line and through a communication link while the credit-card user is purchasing a merchandise. Lots of paper usage is reduced which amounts to less number of trees being cut and eventually a more healthy environment.

8.2 Non-Latin Alphabets

In developing countries, however, there is not much of an effort being done in the field of handwriting recognition. There has been some work on character recognition concerning written languages such as Japanese (Hiragana, Katakana and Kanji) [13, 17], Chinese [13], Korean (Hangul) [46], Hindi (Devanagari) [37], and Arabic [57, 58, 59, 60, 61].

Unfortunately, to my knowledge there has not been any work on Persian (Farsi) or Ordu recognition. To build a recognizer for these languages, lots of effort is needed. Arabic, Persian (Farsi), Ordu and similar written languages are unconstrained by nature. This means that they are the most difficult styles to recognize. A feature of these languages which makes them even harder than English and other Latin style writings is the fact that dots carry a lot of information and based on the number of dots put under or above a basic structure, it could turn into different characters with totally different sounds (see Figure 5). Unfortunately, dots are usually considered to be noise, especially in OCR where they are mostly removed by a filter prior to the recognition. In a language like English, even though i and j have dots, their shapes are distinct and therefore removing the dot will not ambiguate the recognition process.

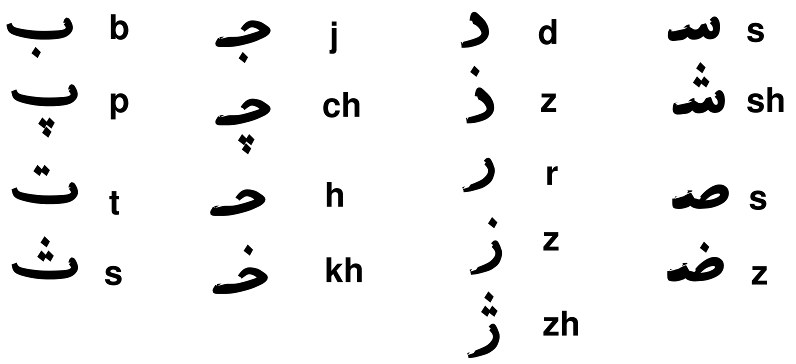

Figure 5: Some Confusing Characters in Farsi and Their Sounds

Due to the ambiguity caused by the importance of noise-like features such as dots, very strong language models are necessary to aid the recognizer. To build such language models, lots and lots of on-line data is necessary such that models of the grammar and word N-grams could be built. There has been an effort in building stochastic models for the Arabic grammar. [59] Similar efforts should be started for other languages such as Persian (Farsi) and Ordu in order to make handwriting recognition possible.

For countries such as Iran that have recently started using computers for common applications, it makes great sense to use pen-computing right from the beginning. Most of the people of these countries, although educated, cannot not type in their own language. Handwriting plays a very important role in these countries versus the western countries. According to a US Postal Service official, 83% of the mail pieces in the United States are typewritten. This figure in a country such as Iran should be very low in my intuition. Handwriting recognition in this sense is an essential simplification tool for computing in countries such as Iran.

References

[1] David Nahamoo and Mostafa Analoui, "Speech Recognition Using Hidden Markov Models," First Annual Conference on Technological Advancement in Developing Countries, Columbia University, New York, June 24-25, 1993.

[2] J. Kim, "On-line Gesture Recognition by Feature Analysis," Proc. of Vision Interface '88, Edmonton, Jun. 6-10, 1988, pp. 51-55.

[3] T.L. Dimond, "Devices for Reading Handwritten Characters," Proc. of Eastern Joint Computer Conference, December 1957, pp. 232-237.

[4] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara, "The State of the Art in On-Line Handwriting Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 8, Aug. 1990, pp. 787-808.

[5] Wacef Guerfali and Réjean Plamondon, "Normalizing and Restoring On-line Handwriting," Pattern Recognition, Vol. 26, No.3, 1993, pp.419-431.

[6] T. Fujisaki, T. E. Chefalas, J. Kim and C. C. Tappert, "Online Recognizer for Runon Handprinted Characters," Proc. of the Tenth International Conference on Pattern Recognition, Atlantic City, New Jersey, June 16-21, 1990, pp. 450-454.

[7] T. Fujisaki, H.S.M. Beigi, C.C. Tappert, M. Ukelson, and C.G. Wolf, "On-line Recognition of Unconstrained Handprinting: a stroke-based system and its evaluation," From Pixels to Features III: Frontiers in Handwriting Recognition, S. Impedovo and J. C. Simon (eds.), Elsevier Publishers, New York, 1992, pp. 297-312.

[8] Tetsu Fujisaki, Krishna Nathan, Wongyu Cho, Homayoon Beigi, "On-line Unconstrained Handwriting Recognition by a Probabilistic Method," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp.235-241.

[9] Shimon Edelman and T. Flash, "A Model of Handwriting," Biological Cybernatics, Vol. 57, 1987, pp. 25-36.

[10] John M. Hollerback, "An Oscillation Theory of Handwriting," Doctoral Dissertation, Massachusetts Institute of Technology, March 1980.

[11] S. Impedovo and J. C. Simon (eds.), From Pixels to Features III: Frontiers in Handwriting Recognition, Elsevier Science Publishers, New York, 1992.

[12] Eveline J. Bellegarda, Jerome R. Bellegarda, David Nahamoo, and Krishna Nathan, "A Probabilistic Framework for On-line Handwriting Recognition," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 225-234.

[13] Kenji Ohmori , "On-line Handwritten Kanji Character Recognition Using Hypothesis Generation in the Space of Hierarchical Knowledge," Pre-Proc. of IWFHRIII 1993, pp.242-250.

[14] Colin A. Higgins and David M. Ford, "On-line Recognition of Connected Handwriting by Segmentation and Template Matching," Proc. of IEEE, 1992, pp. 200-203.

[15] G. H. Abbnik, H. L. Teulings, and L. R. B. Schomaker, "Description Of On-Line Script Using Hollerback's Generation Model," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 217-224.

[16] Adel M. Alimi and Réjean Plamondon, “Performance Analysis of Handwritten Strokes Generation Models,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp.272-283.

[17] Shunji Mori, Kazuhiko Yamamoto, “Research on Machine Recognition of Handprinted Characters,” IEEE Transaction on Pattern analysis and Machine Intelligence, Vol. PAMI-6, No. 4, July 1984, pp. 386-405.

[18] R. Nag, K. H. Wong, and F. Fallside, “Script Recognition Using Hidden Markov Models,” ICASSP, Tokyo, Japan, 1986, pp. 2071-2074.

[19] Fathallah Nouboud and Réjean Plamondon, “On-line Recognition of Handprinted Characters: Survey and Beta Tests,” Pattern Recognition, Vol. 23, No. 9, 1990, pp. 1031-1044.

[20] C. C. Tappert, “Cursive Script Recognition by Elastic Matching,” IBM Journal of Research and Development, Vol.26, Nov. 1982, pp. 765-771.

[21] Yang He, Mou-Yen Chen, and Amlan Kundu, “Handwritten Word Recognition Using Dynamic Segmentation and Hidden Markov Model,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, Vol. I, pp. WA1.25.1-4.

[22] Rabiner, “A Tutorial on Hidden-Markov Models and Selected Applications in Speech Recognition,” Proc. of the IEEE, Vol. 77, No. 2, Feb. 1989, pp. 257-286.

[23] L. E. Baum, “An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes,” Inequalities, Vol. 3, 1972, pp. 1-8.

[24] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang, “Phoneme Recognition Using Time-Delay Neural Networks ,” IEEE Transactions on Acoustic, Speech and Signal Processing, Vol. 37, No. 3, 1989, pp. 328-339.

[25] Lambert Schomaker, “Using Stroke or Character-Based Self-Organizing Maps in the Recognition of Online-Connected Cursive Script,” Pattern Recognition, Vol. 26, No.3, pp. 443-450, 1993.

[26] I. Guyon, P. Albercht, Y. Le Cun, J. Denker, and W. Hubbard, “Design of a Neural Network Character Recognizer for a Touch Terminal,” Pattern Recognition, Vol. 24, Nov. 2, pp. 105-119, 1991.

[27] P. Morasso, L. Barleris, S. Pagliana, and D. Vergava, “Recognition Experiments of Cursive Dynamic Handwriting With Self-Organizing Networks,” Pattern Recognition, Vol. 20, No. 3, pp. 451-460, 1993.

[28] N. J. Nilsson, Problem-Solving Methods in Artificial Intelligence, McGraw Hill Book Company, New York, 1971.

[29] Anna Corazza, Ronato De Mori, Roberto Gretter, and Giorgio Satta, 'Computation of Probabilities for an Island-Driven Parser," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 9, Sep. 1991, pp. 936-950.

[30] Oliviero Stock, Rino Falcone, and Patrizia Insinnamo, “Bidirectional charts: a potential technique for parsing spoken natural language sentences,” Computer Speech and Language, Vol. 3, 1989, pp.219-237.

[31] Homayoon S. M. Beigi, “Character Prediction for On-line Handwriting Recognition,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, Vol. II, 1992, pp. TM10.3.1-4.

[32] Homayoon S. M. Beigi and T. Fujisaki, “A Character Level Predictive Language Model and Its Application to Handwriting Recognition,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, Vol. I, 1992, pp. WA1.27.1-4.

[33] Rohini K. Srihari, S. Ng, Charlotte M. Baltin, and Jackie Kud, “Use of Language Models in On-line Sentence/Phrase Recognition,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 284-294.

[34] Homayoon S. M. Beigi, “A Flexible Template Language Model and its Application to Handwriting Recognition,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, 1992, Vol. I, pp. WA1.28.1-4.

[35] Homayoon S. M. Beigi, T. Fujisaki, W. Modlin, and K. Wenstrup, “A Post-Processing Error-Correction Scheme Using a Dictionary for On-Line Boxed and Runon Handwriting Recognition,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, 1992, Vol. II, pp. TM10.5.1-4.

[36] Radmilo Bozinović and Sargur N. Srihari, “A String Correction Algorithm for Cursive Script Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 6, Nov. 1982, pp. 655-663.

[37] R. M. K. Sinha , “Rulded Based Contextual Post-Processing for Devanagari Text Recognition, “Pattern Recognition, Vol. 20, No. 5, pp.475-485, 1983.

[38] Sriganesh Madhvanath and Venu Govindaraju, “Holistic Lexicon Reduction,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 71-81.

[39] Hendrawa, A. C. Downton, and C. G. Leedham, “A Fuzzy Approach to Handwritten Address Verification,” Pre-Proc. of IWFHRIII, Buffalo, New York, May25-27, 1993, pp. 207-216.

[40] Venu Govindaraju, Ajay Shekhamat, and Sargur N. Srihari, “Interpretation of Handwriting Addresses in U.S. Mail Stream,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 197-206.

[41] Radmilo M. Bozinović and Sangur N. Srihari, “Off-line Cursive Script Word Recognition,” IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 11, No. 1, Jan. 1989, pp.68-83.

[42] Si Wei Lu and He Xu, “Preservation and Recurrency of the Stroke Structure of Characters by Geometric Analysis,” Proc. of Canadian Conference on Electrical and Computer Engineering, Toronto, Canada, Sep. 13-16, Vol. I, pp. WA1.26.1-4.

[43] Hiroyasu Takahashi, “A Neural Net OCR Using Geometrical and Zonal-pattern Features,” ICDAR, 1991, pp.821-828.

[44] Mou-Yen Chen and Amlan Kundu, “An Alternative to Variable Duration HMM in Handwritten Word Recognition,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 82-91.

[45] A. W. Senior and F. Fallside, “An Off-line Cursive Script Recognition System Using Recurrent Error Propagation Networks,” Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp.132-141.

[46] Hee-Seon Park and Seong-Whon Lee, "Off-line Recognition of Large-Set Handwritten Hangul with Hidden Markov Models," Pre-Proc. of IWFHRIII 1993, pp.51-61.

[47] T. K. Worthington, T. J. Chaines, J. D. Williford, and S. C. Gundersen, "IBM Dynamic Signature Verification," Proc. of IFTP, Aug. 1985.

[48] G. Dimanio, S. Impedovo, and G. Pirlo, "A Signature Verification System Based on a Dynamical Segmentation Techniques," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 262-271.

[49] R. H. Anderson, "Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics," Interactive Systems for Experimental Applied Mathematics, M. Klerer and J. Reinfelds (eds.), Academic Press, New York, 1968.

[50] S. K. Chang, "A Method for the Structural Analysis of Two-Dimensional Mathematical Expressions," Information Sciences, Vol. 2, 1970, pp. 253-272.

[51] Z. Wang and C. Faure, "Structural Analysis of Handwritten Mathematical Expressions," Proc. of the 9^{th} International Conference on Pattern Recognition, Nov. 14-17, 1988, Rome, Italy, pp. 32-34.

[52] C. Y. Suen and T. Radhakrishnan, "Recognition of Hand-Drawn Flowcharts," International Joint Conference on Pattern Recognition, 1976, pp. 647-677.

[53] Daniel Lopresti and Andrew Tomkins, "Approximate Matching of Hand-Drawn Pictograms," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 102-111.

[54] Toru Wakahara, "Handwritten Neural Recognition Using LAT with Structural Information," Pre-Proc. of IWFHRIII, Buffalo, New York, 1993, pp. 164-174.

[55] Richard Fenrich and J. J. Hull, "Concerns in Creation of Image Databases," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp. 112-121.

[56] N. Matić, I. Guyon, L. Bottou, and J. Denker, "Computer Aided Cleaning of Large Databases for Character Recognition," International Conference on Pattern Recognition, Amsterdam, Aug. 1992.

[57] A. Namane and M. A. Sid-Ahmed, "Character Scaling by Contour Methods," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.12, No. 6, June 1990, pp. 600-606.

[58] Samir Al-Emami and Mike Urber, "On-line Recognition of Handwritten Arabic Characters," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, July 1990, pp. 704-710.

[59] Ayman El-Naggar, "A Finite State Automata of the Arabic Grammar," Proc. of IEEE, 1989, pp. 693-699.

[60] Adnan Amin and Jean F. Mari, "Machine Recognition and Correction of Printed Arabic Text," IEEE Transaction on Systems, Man , and Cybernetics, Vol. 19, No. 5, Sep. 1989, pp. 1300-1306.

[61] H. Y. Abdelazim and M. A. Hashish, "Automatic Reading of Bilingual Typewritten Text," Proc. of IEEE, 1989, pp. 2.140-2.144.