The *Address*Script™ Recognition System for Handwritten Envelopes

Alexander Filatov, Vadim Nikitin, Alexander Volgunin, and Pavel Zelinsky

ParaScript, LLC, 7105 La Vista Place, Niwot, CO 80503 USA
Tel: (303) 381-3125, Fax: (303) 381-3101,
filatov@parascript.com
www.parascript.com

Abstract. This paper presents *Address*Script - a system for handwritten postal address recognition for US mail. Key aspects of *Address*Script technology, such as system control flow, cursive handwriting recognition, and postal database are described. Special attention is paid to the powerful character recognizer and the intensive usage of context, which becomes available during the recognition process. The algorithm of confidence level calculation is presented. Laboratory test results on a blind test set of 50,000 images of live hand-written mail pieces demonstrate a 64% finalization rate for error rates below USPS restrictions.

1 Introduction

Mail sorting is one of the most valuable applications of document analysis and handwriting recognition. About 36 million of handwritten envelopes are processed daily by the United States Postal Service (USPS). The USPS uses the Remote Computer Reader (RCR) developed by Lockheed Martin Postal Systems to process handwritten and machine print mail pieces. Current recognition system deployed throughout the United States last year provides about 25% read rate [1]. Lockheed Martin Postal Systems continues to improve the RCR system. Within this work they have successfully integrated the Parascript *Address*Script product into the RCR. Subsequent tests have demonstrated a significant increase in the RCR read rate. USPS has contracted with Lockheed Martin Postal Systems to deploy the new RCR version with the integrated *Address*Script product throughout the United States.

For mail-sorting purposes all information presented in the destination address block of each mail piece is encoded by a string of digits (up to eleven digits) called Delivery Point Sequence Code (DPS). The first five digits are referred to as the ZIP Code. A ZIP Add-On Code is a four-digit number that follows the five-digit ZIP Code. The five-digit ZIP Code divides the U.S. into major geographical areas. The ZIP Add-On Code identifies a small geographical delivery area that is serviceable by a single carrier. The last two digits of the DPS uniquely identify each carrier delivery point. The Five-digit ZIP Code is usually written on an envelope. The ZIP Add-on

S.-W. Lee and Y. Nakano (Eds.): DAS'98, LNCS 1655, pp. 157-171, 1999.

Code may be extracted from the USPS Delivery Point File for a given ZIP Code, street number and street name. The last two digits of the DPS generally represent the last two digits of a street number.

The interpretation of hand-written addresses is a difficult problem of document analysis and handwriting recognition. Addresses may be written in different styles: city, state, and ZIP Code may occupy one, two, or three lines; the ZIP Code may contain five or nine digits; the last line of an address block may contain other information (e.g. "Attn.", "USA" etc.); street address may occupy one or two lines. There are several different address structures: street, PO Box, rural route addressing, etc. Very often an address is written in cursive so the handprint recognition technology alone can not meet the needs of address interpretation. Figure 1 shows examples of an address block.

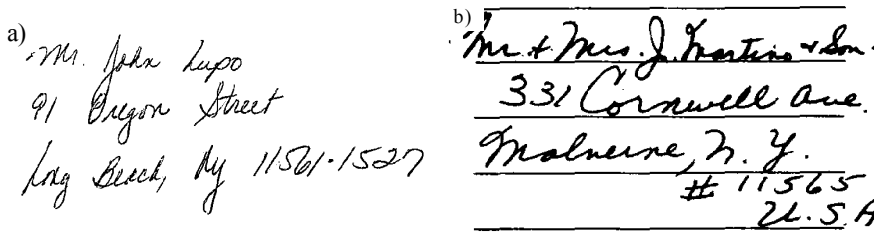

Fig. 1. Examples of address block images.

*Address*Script is a system intended for the recognition of handwritten U.S. postal addresses. Significant increase in the finalization rate compared to the published result [1] is achieved due to the following major factors:

*Powerful recognition engine***.** Our Script Recognizer combines the Holistic Word Recognizer (HWR) intended for cursive recognition and the Analytical Word Recognizer (AWR) intended primarily for recognition of numbers and hand-printed words. AWR was tested on hand-written numerals contained in the CEDAR database [2] and achieved a 99.54% recognition rate, which is the highest rate ever reported on this database.

*Extensive use of the context, which becomes available during the recognition***.** The idea was to adapt the system to an address being recognized. For example, street address block parsing is performed after ZIP Code recognition since the ZIP Code recognition results provide additional information, which can be used for street address parsing. This information includes the percentage of each address type (street, PO Box, rural route), minimum and maximum street number length for ZIP Code, etc.

*Reliable determination of the recognition answer correctness***.** Neural network is used to calculate the confidence level for a particular answer. The neural network takes the information collected at all stages of the previous recognition process as input and outputs the probability of the answer correctness.

2 *Address*Script Recognition Technology

It is necessary to have a powerful recognition engine to deal with real-life handwriting. The core of the *Address*Script technology is a Script Recognizer (Figure 2). It is a combination of two recognizers: Holistic Word Recognizer [2] and Analytical Word Recognizer [3].

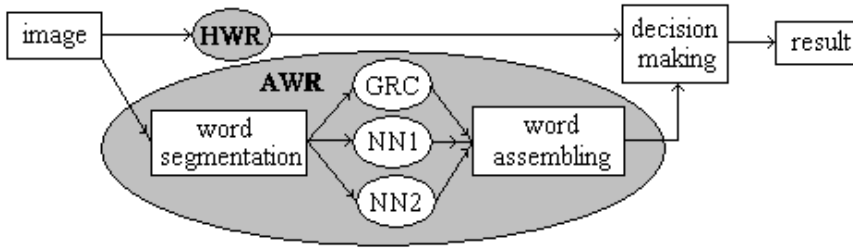

Fig. 2. The structure of the Script Recognizer.

2.1 Holistic Word Recognizer

The strength of the HWR is the ability to recognize cursive handwriting. It can also recognize hand printed information or a combination of handwriting and hand print.

The input to the HWR is an image of a whole word or a phrase written in one string, and a lexicon. The output is a list of probable answers with their respective similarity scores.

The recognition approach can be classified as a holistic one. This means that a word is recognized as a whole without segmentation to the character level. Recognition consists of two stages: a) feature extraction and feature ordering in a linear sequence, b) matching the feature representation of an input phrase to all lexicon entry representations.

The set of features reflects the importance of vertical extremums, which are the most stable elements of writing. It includes eight features to describe maximums (and corresponding eight features for minimums): *arc*, *arc with a right (left) free end, cusp*, three *long cusps* (upper, upper-right, upper-left), *loop*. Also we use *cross*, *dot*, *dash*, *right (left) arrow*, and several features to describe holes of different shapes (the feature set is described in detail in [2]). Every feature has a height attribute determined by the vertical location of the feature relative to the guidelines (the body of the word). We introduce a measure of similarity between different types (and different heights) of features. This provides a high flexibility in the matching process necessary for the recognition of sloppy handwriting.

Each lexicon entry representation is generated dynamically via concatenation of predefined letter representations. Letters are described by a linear sequence of features. Each letter can have several feature representations (four on the average), so

each lexicon entry has a lot of feature representations. The length of the sequence may vary from one feature per letter up to ten features.

Dynamic programming based on a string edit distance is used to match an input word representation with all lexicon entry representations.

2.2 Analytical Word Recognizer

The Analytical Word Recognizer is primarily intended for the recognition of numbers and hand-printed words. AWR has two stages: segmentation and recognition. At the segmentation stage a graph describing the shape of an input word is built. Then the graph is segmented into separate symbols. Obviously, it is not always possible to find the correct segmentation variant without recognition. Therefore, we use a multi-variant segmentation approach. A thorough analysis of the segmentation variants before recognition allows consideration of less than two variants per symbol on the average. After that the recognizer determines the best alternative.

Table 1. Recognition rate on *goodbs* data of the CEDAR database for different error rates.

Error rate	Recognition rate
0.0	93.80
0.05	96.24
0.10	98.14
0.20	99.18
Without Reject	99.54

Table 1 shows AWR test results on handwritten numerals from the CEDAR database. These data were collected from live mail in the U.S. [4]. The directory *goodbs***,** containing 2213 samples, is widely used to compare different recognition systems. We did not use these samples for training. AWR obtained 99.54% recognition rate, which is the highest rate ever reported on this database (Table 2).

Table 2. Published systems (recognition rate without reject).

T.M. Ha, H. Bunke [5]	99.09
D.S. Lee, S.N. Srihari [6]	98.87
S. Knerr, S. Sperduti [7]	97.6

Three classifiers inside AWR perform the symbol recognition stage: graph-based character classifier (GRC) [3] and two neural network classifiers based on different features.

2.3 Graph-Based Character Classifier

GRC recognizes input symbols involving an algorithm of matching input symbol graphs with previously defined symbol prototypes. The symbol prototype consists of the symbol graph and the description of its elements (geometrical characteristics of edges, mutual position of edges and nodes, etc.). An obligatory subgraph is defined in each prototype graph. For each element of the obligatory subgraph a match should be found in the input graph.

Each input symbol graph may contain elements that have their analogs in the prototype and the extra elements. The input symbol graph is transformed according to prototype rules to reduce extra elements. The transformations are fulfilled until a one-to-one match is achieved between the input graph and the prototype graph or its obligatory subgraph.

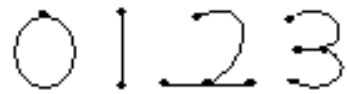

Fig. 3. Examples of symbol prototypes

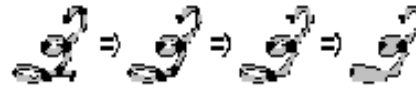

Fig. 4. The sequence of transformations of an input character graph during the matching process.

The estimate of a match between a transformed input graph and the prototype graph is based on the descriptions of the prototype elements and allows certain differences between an original input graph and a prototype.

2.4 Neural Network Classifier

The Neural Network (NN) classifier is intended for recognition of a segment, which is expected to be one character or a few characters written as a whole.

Neural networks are used at the segmentation stage, for character recognition, and for the calculation of the final confidence level for an answer. *Address*Script uses six neural networks: 2 – for digit classification, 2 – for letter classification, 1 – to discriminate between letters and digits, 1 – for confidence level calculation. Here we will discuss the neural network classifiers used for character recognition. To improve the recognition performance *Address*Script uses several NN classifiers to recognize the same character. These NN classifiers use different feature representations of the image as input. Then, the output of the NN classifiers is analyzed, and vector of

probability estimations for each class is calculated. The combination of classifiers based on different features reduces the classification error rate because such classifiers rarely produce same errors on the same image.

Our work on NN classifiers has its origin from the NN classifier initially developed by I. Lossev, N. Bagotskaya and coworkers from Paragraph International.

Feature Extraction. We use two main types of features for the character classification.

The first type of features is Fourier descriptors. First, we represent the image as a two dimensional curve. It can be either the contour or the skeleton of an image. Then, the curve is smoothed, and the descriptors are calculated as Fourier coefficients of the x(t) and y(t) – parameterized representation of the curve. We use a different number of coefficients (20-30) depending on the recognition task.

To build the second type of features we split the normalized box of a symbol into 3x3 evenly distributed cells. For each point of the image border within a cell the tangent vector is calculated. Then, vectors are projected on four directions (vertical, horizontal, and two diagonals) and the sums of absolute values of projections over the points of the border form four features corresponding to the cell. So, the total number of features in this case is 36.

Structure of the Neural Network. We use Radial Basis Function (RBF) neural network with one RBF layer for classification:

$$OUT_i = 1 - \prod_{j=1}^{N_i} \left(1 - \exp\left\{ - \frac{\left(\mathbf{A}_{ij}^T \mathbf{A}_{ij} \left(\mathbf{x} - \mathbf{c}_{ij} \right), \left(\mathbf{x} - \mathbf{c}_{ij} \right) \right)}{r_{ij}} \right\} \right) \quad \textbf{(1)}$$

where: $\mathbf{x}$ - input vector of features, OUT_i - output weight of class i, N_i - number of RBF neurons for class i, $\mathbf{A}_{ij}$ - weight matrix of j-th neuron of class i, $\mathbf{c}_{ij}$ - center of j-th neuron of class i, and r_{ij} - radius of j-th neuron of class i.

We chose this structure of the neural network based upon the following considerations: The coefficients of this neural network have a clear 'physical' meaning. Vector $\mathbf{c}_{ij}$ corresponding to the neuron can be treated as a vector of features of some 'center' image of the neuron and can be easily visualized. 'Radius' r_{ij} of the neuron shows how 'far' the image can be from the 'center' to still be classified as belonging to the class i. Eigen vectors and corresponding eigen values of the matrix $\mathbf{A}_{ij}^T \mathbf{A}_{ij}$ show the 'directions' and corresponding 'distances' of admissible deviations of the image of the same class from the 'center'. So, this neural network is not a 'black box', and in some cases one can understand how and why one or another result of the classification is produced. We developed special tools for the visualization of the NN.

The proposed structure of the neural network allows us to choose reasonable initial values to the NN coefficients before learning (See more detailed description in the next section). As one can see, the outputs for each class are independent. So, the corresponding parts of the NN can be learned separately. Also, it simplifies the process of tuning of the NN for a new application. Thus, if you need to add one more

character class to existing NN it is enough to learn only part of the NN corresponding to the new class. For example, it is necessary to add '#' class to a digit classifier to recognize an apartment number on an envelope successfully.

In some cases, we use modifications of the described structure of NN classifier. For example, matrices $\mathbf{A}_{ij}$ can be unary or diagonal. Such NN classifiers are used when the constraints on the speed of recognition and the memory used by the classifier are more important than the performance of the classification.

Initial Classification. Our experience shows that the choice of initial values of NN coefficients is very important to achieve high performance of the NN classifier. We call this process initial classification. The aim of initial classification is to build neurons ($\mathbf{c}_{ij}$, r_{ij} , $\mathbf{A}_{ij}$) which produce reasonable output on most of the samples in the learning set. We tried several algorithms of initial classification, but now only two of them are used. Both algorithms consist of two stages.

At the first stage each sample of the class is considered as a candidate to be a neuron 'center' and corresponding neuron coefficients ($\mathbf{c}$, r , $\mathbf{A}$) are chosen. Let us call samples of the same class native samples of the center candidate and the rest samples the alien ones. The value of $\mathbf{c}$ is always set to the feature vector of the sample. In the first algorithm (isotropic classification), the matrix $\mathbf{A}$ is unary and r is equal to the distance to the nearest alien sample. In the second algorithm (ellipsoidal classification), r is unit but the matrix $\mathbf{A}$ is chosen so that the axes of the ellipsoid $(\mathbf{A}^T\mathbf{A}(\mathbf{x}-\mathbf{c}),\mathbf{x}-\mathbf{c})=1$ have maximal length provided that all alien samples were situated outside the ellipsoid.

At the second stage, we choose initial neurons among the built candidates. Each neuron candidate has a set of 'covered' native samples (those being inside the ellipsoid/sphere $(\mathbf{A}^T\mathbf{A}(\mathbf{x}-\mathbf{c}),\mathbf{x}-\mathbf{c})=r$). The target of the procedure is to choose the candidates which 'cover' the maximal number of native samples. Thus, the problem of the selection of the best center candidates is reduced to the mathematical problem of the selection of the best cover of the set by a predefined number of subsets from a given set of subsets. We use well-known 'greedy' heuristics to find the approximate solution of the problem.

The final performances of NN classifiers built using ellipsoidal classification are close to that of classifiers, based on the isotropic classification. However, ellipsoidal classification is favorable to build neurons corresponding to the rare ways of symbol writing.

Neural Network Learning. We use a gradient relaxation method to learn our NN classifiers. We split the set of samples used for NN learning into a 'train' set, used for gradient relaxation, and 'test' set, used to monitor the performance of the classifier during learning. After each epoch of gradient relaxation, the classifier is tested on the 'test' set. If during some predefined number of epochs, the state of the classifier remains worse than that in the best state, learning is completed. Using the 'test' set eliminates over-learning of the classifier on the 'train' set with the corresponding loss of the generalization ability.

3 System Control Flow

The input of the *Address*Script system is a binary image of an envelope (or an address block). The output is an answer and a confidence level for ZIP5 Code and for the ZIP Add-On Code. Additional output information includes an answer, a confidence level, and field location coordinates for each recognized field: city, state, street number, street name, etc. The overall *Address*Script system control flow is shown in Figure 5.

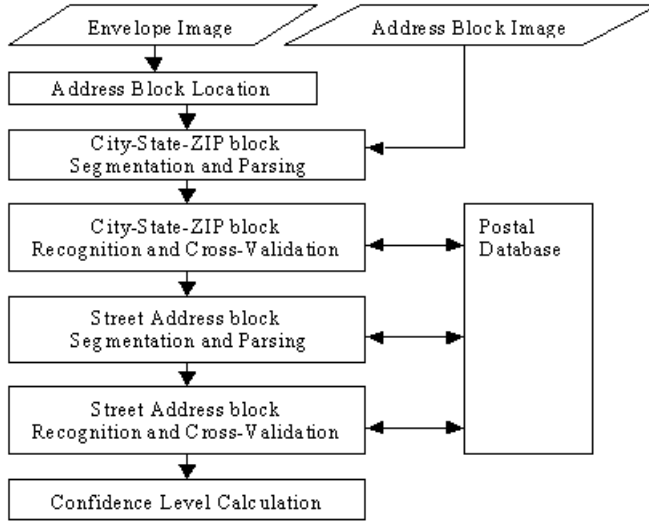

Fig. 5. Overall Control Flow

3.1 Postal Database

Postal database used by *Address*Script system is generated from the USPS City-State and Delivery Point files. Original text files were compacted, and their size was reduced from 15 GB to 670 MB.

The database is queried with a ZIP Code to retrieve all (City, State) pairs; with a ZIP Code and a street number to retrieve all street names and apartment numbers; with a ZIP Code to retrieve all PO Box and rural route numbers.

Aliases of cities (Los Angeles – LA), states (California – CA – Cal), street names, suffixes (Road - Rd), pre- and post-directions (North - N), PO Box and rural route names are stored in the database. The database also contains pre-calculated information about the presence of PO Box, street and high-rise type of addresses in the given ZIP Code, street and PO Box number minimal and maximal values, presence of alphanumeric street numbers and names, etc. This information is used to improve street address block parsing, and to restrict the context for street address recognition.

3.2 City-State-ZIP Block Processing

Our City-State-ZIP block-processing algorithm is described in detail in [8]. Segmentation and parsing module accepts lines of an address block as input. Up to three bottom lines are processed. First, each line is divided into connected components. Next, the algorithm combines all connected components into clusters according to the size, location, and distance between neighbouring components. Clusters are classified as a word, group of characters, separate character (letter or digit), separate letter, separate digit, comma, dash, or noise. We use a neural network, which discriminates between letters and digits at this stage. The parsing algorithm takes a list of clusters as input and provides a list of hypotheses for the city, state, ZIP Code, and 4-digit ZIP Add-On Code. Up to four hypotheses are generated and scored depending on the image complexity. If it is impossible to separate city from state unambiguously, the algorithm composes a long word which includes both city name and state name candidates to reduce the number of segmentation variants. Such a word is recognized with a combined lexicon that consists of city-state pairs.

The correct hypothesis is among the candidates in 91% of the cases. If the correct hypothesis is in the list, it ranks first in 90% of the cases and ranks second in 7% of the cases. Frequently, there is no need to process all segmentation variants. Hypotheses with higher parsing scores are processed first. If the confidence level of a recognized ZIP code is high enough for some hypothesis, the procedure is terminated; otherwise the next segmentation hypothesis is investigated.

The recognition process starts with the ZIP Code. A lexicon of about 42000 ZIP Codes is obtained from the postal database. The algorithm produces a list of ZIP Code candidates (up to 30 answers).

Next, the recognition of city is performed. The postal database is queried by each ZIP Code in the list to obtain a list of city names. Retrieved city names along with their aliases form a lexicon (for example, “MOUNTAIN HEIGHTS”, “MTN HEIGHTS”, “MOUNTAIN HGTS”, and “MT HEIGHTS”). State names and their aliases are also added to the lexicon. After the recognition, the aliases are converted to the standard city name. If the best answer is a state name, the city is considered to be on the previous line. In this case additional parsing hypothesis with city name on the previous line is generated and processed.

After recognition, the city and ZIP Code answer lists are cross-validated using the postal database. A score is calculated as a weighted sum of the city and ZIP Code scores for each valid (city, ZIP Code) pair. The score of the best pair represents the final score of the parsing hypothesis.

The processing of the City-State-ZIP parsing hypotheses is terminated if the score is high enough; otherwise the next hypothesis is processed. The hypothesis with the highest score produces the final result of the City-State-ZIP block recognition.

3.3 Generation of ZIP Code List for Street Address Processing

Street address processing is based on the results of City-State-ZIP block recognition. The algorithm chooses several ZIP Code hypotheses to build a lexicon for street name recognition. The list of ZIP Codes, which is used for further processing includes:

- Several top choices of the ZIP Code obtained after the cross-validation with city recognition results.
- The best ZIP Code hypothesis obtained as a result of the ZIP Code field recognition (before the cross-validation with city recognition results) if its score is significantly higher than the score of the best cross-validated ZIP Code.
- When the ZIP Code has been changed (translated) a writer may still use the old ZIP Code to address a letter. Information concerning ZIP Code translations is stored in the database. There may be several new ZIP Codes instead of the original one. If a ZIP Code in the list is marked as translated, we add a corresponding new ZIP Code to the list.

Up to 5 Zip Codes can be used to build a lexicon for street name recognition.

3.4 Street Address Processing

The street address segmentation and parsing module takes the list of address block lines as input. Up to two lines above the top line of the best City-State-ZIP hypothesis are processed. The street address parsing is based upon the same principles as City-State-ZIP block parsing. The difference is that it uses the postal database information about the presence of PO Boxes, street addresses, and apartments in the list of ZIP Codes, as well as minimal and maximal length of street, apartment and PO Box numbers. The parsing algorithm generates up to four segmentation hypotheses depending on the image complexity. The hypotheses can be one of the following types: PO Box or rural route, Street address, and Street address with apartment.

PO Box or rural route hypotheses consist of name and number components. Street address hypotheses consist of street number and street name components. Hypotheses of a street address with apartment have an additional apartment number component.

The recognition of street numbers, apartment numbers and PO Box / rural route numbers is performed by the Analytical Word Recognizer. Minimal and maximal possible answer values for given ZIP Codes are retrieved from the postal database and used during the recognition. The recognition of street name and PO Box / rural route names is performed by the Script Recognizer and it is lexicon based.

The recognition of PO Box / rural route hypothesis starts with the name component. In this case HWR uses 3 separate lexicons: PO Box names, apartment abbreviations, and rural route names. After the recognition, the best answer is considered. If it came from the lexicon for apartment abbreviations the hypothesis is converted to a street address with an apartment number, otherwise the numeric component is processed.

After the numeric component recognition, the postal database lookup is performed and valid PO Box / rural route answers compose the answer list. The scores of the answers are calculated as a weighted sum of scores of the name and the number.

The processing of street address and street address with apartment number hypotheses starts from the recognition of the street number component. Next, the lexicon of street names is generated. To create the street name lexicon, the postal database is queried with ZIP Code and street number. Depending on the street number confidence level, the postal database is queried either with all street number answers

or with the best answer only. The database contains street names in a standard form with abbreviated suffixes, pre- and post-directions (for example, "W OAK DR"). A writer may use non-abbreviated forms, omit or mix up some parts of a street name. Therefore we add to the lexicon:

- Street names with non-abbreviated suffix, pre- and post-directions ("W OAK DRIVE", "WEST OAK DR", "WEST OAK DRIVE")
- Street names without suffix or directions ("W OAK", "WEST OAK", "OAK DR", "OAK DRIVE")
- Street names with substituted suffixes (only frequent replacements such as: Avenue - Street, Drive – Street - "W OAK STREET", "WEST OAK STREET")
- Reversed pre- and post-directions ("OAK DR W", "OAK DRIVE W", "OAK DR WEST", "OAK DRIVE WEST")

Penalties are applied to a score of an answer if the recognition answer is one of the following: street name without suffix, with substituted suffix, with reversed pre- or post-direction. After the recognition, the street aliases are converted to the standard street name.

Next, the apartment number component is processed.

After the recognition of all components, the postal database lookup is performed, and recognition results are cross-validated using the weighted sum of component scores. The hypothesis with the highest score produces the final result of address recognition.

If a writer specifies a 9-digit ZIP Code, 4-digit ZIP Add-On Code is recognized and cross-validated with the ZIP Add-On Code retrieved from the database using the street address recognition results. In real life a writer often applies a wrong ZIP Add-On Code. To handle this situation we perform cross-validation only if a ZIP Add-On Code is found both in the street address answer list and in the digit ZIP Add-On Code answer list. Otherwise, 4-digit ZIP Add-On Code written on the envelope is ignored.

4 The System Adaptability

In this system we tried to use the context, which becomes available during the recognition in the most efficient way. The idea was to adapt the system to the address being recognized. What does this mean? There are two possible sources of information. First, when we generated our Postal Database we accumulated various data available for every ZIP5 Code. This information includes:

- Percentage of each address type: PO Box, street address, street address with secondary information (apartments), rural route, etc. It is used for street address block parsing to generate (or not to generate) some doubtful parsing variants.
- Minimal and maximal street number and PO Box number. They are used for street address block parsing and for street number (PO Box number) recognition (both at segmentation and recognition stages).
- Presence of an alphanumeric street number, street name, or PO Box number. This information should be taken into account to avoid mistakes in street address block

parsing. At this stage we use a special neural network, which has been trained to discriminate between letters and digits. In the case of an alphanumeric street number, street name, or PO Box number it is necessary to be careful about interpreting the neural network results.
- This information is also used to apply appropriate neural networks for the recognition.
- Percentage of street names with and without pre-directions, post-directions. It is used for street address block parsing.

Second, we accumulate information concerning peculiarities of a given address at each recognition stage and then we try to use it in subsequent stages. Let us consider two examples:

- Street suffix absence. The analysis is conducted during the street address block parsing and the information is passed to the monitor program. It influences the lexicon for street name recognition. There are three possible solutions depending on this information: to include street names with and without suffix in the lexicon, to include only street names with a suffix, to include only street names without a suffix.
- An apartment number written on a separate line. Sometimes people write a street address in two lines with apartment number written on the second line. It is difficult to parse this street address correctly because it looks like a PO Box address (a name, and after that a number). In this case, recognition can help to detect the parsing mistake. We add apartment names (apartment, apt. suite, #, etc.) into the lexicon for PO Box name and rural route name reading. If after PO Box name recognition the best answer is one of the apartment names, the street address parsing is called again with the corresponding information.

5 Confidence Level Calculation

It is important to get a reliable estimate of answer correctness. Current restrictions of the USPS on error rate are as follows: five-digit ZIP Code error rate should be less than 1%, and ZIP Add-On Code error rate should be less than 2%. The error rate is calculated as a ratio between the number of accepted wrong answers and the total number of accepted answers.

We need a specific reject procedure to achieve the highest possible accept rate provided that the error rate is lower than a priori defined value. It is not convenient to have an output of the reject procedure in the form of ‘accepted’ or ‘rejected’ because it demands that the reject procedure be tuned to each specific error rate. To solve this problem, the recognizer outputs a confidence value associated with each recognition answer. If it is necessary to provide a certain error rate, one should choose an appropriate threshold and consider the answers with the confidence levels higher and lower than the threshold as accepted and rejected respectively.

We use the neural network described above to calculate the confidence level of an answer. The neural network takes the information collected at all stages of the previous recognition process as input. It includes:

- The confidence of address block location
- The confidence of city-state-ZIP block and street address block parsing
- The confidence of recognition of each specific field (city, ZIP5, ZIP Add-On Code, street number, street name, etc.)
- The confidence of ZIP5 calculated after cross-validation of city field and ZIP5 field recognition results
- The confidence of the whole street address block calculated after cross-validation between street number and street name recognition results (or PO Box number and PO Box name, etc.)
- The difference in scores between the final answer and the best answer from all the other segmentation variants for the street address block
- Street address type (PO Box, rural route, street address, street address with secondary information)

The length of the input vector is 14. Output is the only value – probability of the answer correctness. Sample set size is 75,000 input vectors (generated in the process of recognition). It is divided in two parts: 65,000 - training samples, 10,000 - verification samples (see neural net learning details above). The confidence level of each specific field is an output from the Script Recognizer. The procedure of its calculation is described in [9].

6 Performance

The *Address*Script software has been implemented in the C programming language and developed under the Microsoft Windows NT environment. The size of the executable code is less than 4 MB. The size of the running code fits 32 MB. The average processing time of one envelope is less than 750 ms on a Pentium II – 300 MHz.

Table 3. Performance of the AddressScript system at various confidence level thresholds.

Finalization rate	Five-digit ZIP error	ZIP Add-On error	DPS error
64.0	0.8	2.0	0.2
61.8	0.7	1.5	0.2
56.5	0.5	0.9	0.1
51.0	0.4	0.5	0.1

The laboratory testing of the system was conducted on a blind test set of 50,000 images of live hand-written mail pieces (Table 3). The system has achieved a 64% finalization rate (finalization rate may be lower under real life operation). This result compares favorably with 25% finalization rate reported in [1]. Finalization rate means the percentage of accepted images (all the other images should be sent to an operator for manual treatment). The error rate is calculated as the number of errors among

finalized images divided by the total number of finalized images. It is calculated separately for the five-digit ZIP Code, ZIP Add-On Code, and the last 2 digits of the DPS Code. The reason is that the costs of these errors are different. The most expensive is five-digit ZIP Code error because in this case the letter will be sent far from the correct destination. ZIP Add-On Code error is cheaper than five-digit ZIP Code error and the cheapest is a DPS error. If several errors occur simultaneously, only the most expensive one is taken into account. Current USPS error rate restrictions are: less than 1% for ZIP5 errors, and less than 2% for ZIP Add-On Code errors.

7 Conclusions

We have described *Address*Script - a system for handwritten postal address recognition for US mail. The average processing time of one envelope is less than 750 ms on a Pentium II – 300 MHz. The system has achieved a 64% finalization rate while being tested on a blind test set of 50,000 images of live hand-written mail pieces. The corresponding Five-digit ZIP error rate is 0.8%, ZIP Add-On error is 2.0%, and DPS error is 0.2%. These error rates are below current USPS requirements.

Significant increase in finalization rate compared to the published result [1] is achieved due to the following major factors:

- Powerful recognition engine, based on the combination of the Holistic Word Recognizer intended for cursive recognition and Analytical Word Recognizer intended primarily for number recognition and hand-printed word recognition.
- Extensive use of the context, which becomes available during the recognition**.**
- Reliable estimation of answer correctness based on the information collected at all stages of the previous recognition process.

New RCR systems with the integrated *Address*Script product were deployed in more than forty USPS sites throughout the United States by the end of August 1998. It is scheduled to deploy more than 160 new RCR systems by the end of 1998.

References

1. Srihari, S.N. and Kuebert, E.J.: Integration of Hand-Written Address Interpretation Technology into the United States Postal Service Remote Computer Reader System. Proc. of the 4th Int. Conf. on Document Analysis and Recognition. Ulm, Germany. (1997) 892-896

2. Dzuba, G., Filatov, A., Gershuny, D., and Kil, I.: Handwritten Word Recognition – The Approach Proved by Practice. Proc. of the 6th Int. Workshop on Frontiers in Handwriting Recognition. Taejon, Korea. (1998) 99-111

3. Filatov, A., Gitis, A., and Kil, I.: Graph-Based Handwritten Digit String Recognition. Proc. of 3rd Int. Conf. on Document Analysis and Recognition. Montreal, Canada. (1995) 845-848

4. Fenrich, R. and Hull, J.J.: Concerns in Creation of Image Database. Proc. of the 3rd Int. Workshop on Frontiers in Handwriting Recognition. Buffalo, New York, USA, May 25-27, (1993) 112-121
5. Ha, T.M. and Bunke, H.: Handwritten Numeral Recognition by Perturbation Method. Proc. of the 4th Int. Workshop on Frontiers in Handwriting Recognition. Taipei, Taiwan, Republic of China. Dec. 7-9 (1994) 97-106
6. Lee, D.S. and Srihari, S.N.: Handprinted Digit Recognition: A Comparison of Algorithms. Proc. of the third Int. Workshop on Frontiers in Handwritting Recognition. Buffalo, New York, USA. May 25-27 (1993) 153-162
7. Knerr, S. and Sperduti, S.: Rejection Driven Hierarchy of Neural Network Classifiers. Proc. of 1993 Int. Symp. on Nonlinear Theory and its Applications. Vol. 3. Hawaii. (1993)
8. Dzuba, G., Filatov, A., and Volgunin, A.: Handwritten ZIP Code Recognition. Proc. of the 4th Int. Conf. on Document Analysis and Recognition. Ulm, Germany (1997) 766-770
9. Dzuba, G., Filatov, A., Gershuny, D., Kil, I., and Nikitin, V.: Check Amount Recognition based on the Cross Validation of Courtesy and Legal Amount Fields. In: Impedovo, S., Wang, P.S.P., Bunke, H. (eds.): Automatic Bankcheck Processing. World Scientific. (1997) 177-193