A Multi-level Pattern Matching Method for Text Image Parsing

Michal Prussak and Jonathan J. Hull

Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260
prussak@cs.buffalo.edu, hull@cs.buffalo.edu

Abstract

A multiple level pattern matching approach to text image parsing is described. The parser assigns syntactic categories to words that occur in a two-dimensional image of a constrained block of text. Bottom-up information (from an image segmentation routine), that includes the number of lines of text, the number of words in each line, and an estimate of the number of characters in each word, is initially used to compute a ranked list of categories for each word and a probability that each one is correct. The local support provided by words that are horizontally adjacent is used to refine the initial assignment and to assign a ranked list of categories to each line. The word and line category assignments are then probabilistically fit to patterns that describe allowable configurations of lines. The output is a ranked list of the patterns that fit best and the probabilities that they are correct. A success rate of up to 96 percent is achieved in classifying lines of text in a test set of postal address images.

AI topic: Knowledge-based control, parsing.
Domain area: Process automation.
Language/Tool: Lisp, C, and Unix.
Status: Initial development complete. Being tested in a research system for knowledge-based postal address understanding.
Effort: 1.0 person-years.
Impact: Estimated benefit is a significant increase in the ability to accurately assign categories to words in postal addresses.

1. Introduction

Image parsing is the process of assigning categories to segmented regions in an image. The categories are high level descriptions of the contents of the regions and are used to guide subsequent recognition. Aerial image understanding systems often contain a parsing step [5]. An example would be in processing a complex image of a city. Parsing would label the segmented regions as a "building", "car", "road", and so on. A separate recognition process would identify the specific building, car, or road present in the scene. Another example of image parsing occurs in many artificial intelligence based systems for document understanding [1,4]. Regions are segmented and identified as graphics, text, line drawings, and so on before recognition of the text or other regions is performed. In such approaches, the importance of accurate parsing is obvious. The success of later recognition stages and thus the overall interpretation of the image depends on the success of parsing.

The parsing of images of text can be approached as a problem of normal ASCII text parsing where the identities of the words are known accurately and the objective is to assign categories such as *noun*, *verb*, and so on, to words [11]. Such methods typically are provided with a set of rules about configurations of categories and dictionaries that associate words with their possible categories. An application of this methodology to text image parsing used an augmented transition network (ATN) grammar that described two-dimensional relations between words in an image [7]. (The grammar was represented and applied using the SNePS semantic network processing system [9].) Such an approach is successful if the identities of words are known before parsing and if a comprehensive dictionary is available. However, this information may not always be present because the recognition of word images is an imperfect process [6] and words can appear in an image that are not in a dictionary. An alternative would be to use methods for parsing ill-formed input that could account for variations in input quality [2]. However, similar problems of word recognition and dictionary generation would persist.

Pattern matching methods for parsing are applicable to constrained domains where they can describe the variations that occur. The most famous pattern matching method is the ELIZA program that simulated a psychiatrist interviewing a patient [10]. A set of keywords and phrases were matched against an input phrase and responses were generated based on the matches that occurred. This methodology has also been successfully utilized in other natural language processing systems for constrained domains such as PARRY that simulated a paranoid patient [8], and FRUMP that summarized newspaper stories [3].

A pattern matching approach to text image parsing could avoid word recognition and dictionary generation problems by not using word identities. Instead, a large collection of patterns could be used to describe the configurations of image data that could occur. Such a system could be easily re-trained by giving it a new set of patterns. This would avoid the time consuming process of writing a new rule base that would occur if a parser like an ATN were used. A pattern matching text parser is feasible if information other than recognition results constrain the identity of words. Such information is present in a domain like postal addresses where the two-dimensional arrangement of words is informative and other data such as the number of characters in a word is meaningful. For example, a two-letter word in the first position of the top line in a three line address is very likely to be a person's title (Mr, Ms, Dr, and so on). Furthermore, this domain is sufficiently constrained so that all the patterns of word classes can be listed.

The remainder of this paper presents a pattern matching approach to two-dimensional text image parsing of postal addresses. The implementation of the algorithm and the results of testing are also discussed.

2. Algorithm Description

The algorithm for text image parsing that is applied to postal addresses is illustrated in Figure 1. An address image that has been segmented into lines and words is input. The step of initial word class assignment uses the following information to assign a list of potential categories to each word:

1. Number of lines of text in address;
2. Line number on which the word is located;
3. Number of words in the line in which the word is located;
4. Position of the word in the line;
5. Number of characters in the word;
6. Whether the word is composed of letters or digits.

Note that no explicit recognition results for the text are used. The output of this step is a list of potential categories for each word and the probabilities that they are correct.

The horizontal refinement stage parses each line individually. The initial categories assigned to each word and their probabilities are fit to the horizontal patterns that could apply to this line. The output is a ranked list of the classifications for each line and the probabilities that the line classifications are correct.

The vertical refinement stage determines the parse for the entire address block. The classifications for each line and their probabilities are fit to a given set of patterns that describe the allowable vertical arrangements of lines within an address block. The output of this stage contains a ranked list of vertical patterns and the probabilities that they are correct. These describe several alternative parses for the address block. Re-ordered lists of the configurations of words for each line that are consistent with the vertical patterns are also output.

2.1. Initial Word Class Assignment

The first step of the parser returns a list of possible classes for each word in an address block. The lists are ordered by the probabilities that the classes are correct. Overall, 41 classes such as *street number, pre-*

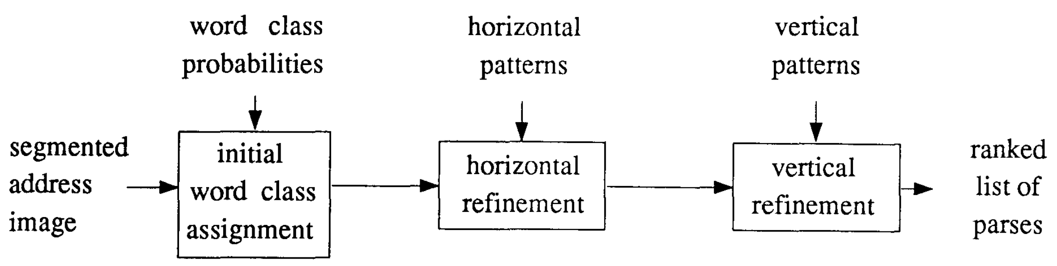


Figure 1. Overall algorithm design

directional, firm name and ZIP Code are used by the system. The list of classes and their confidences are calculated by indexing a data structure to retrieve the frequency of occurrence of each class in a given context. These frequencies are then used to calculate a weighted confidence that each class is correct.

A four level tree data structure holds the basic frequency information. A path from the root to a terminal node is described by a combination of four of the six items of information listed earlier (number of lines in the address, line number of the word, number of words in that line, and the number of the word). The terminal nodes in the tree are frequencies of word classes with different numbers of characters.

An example of a portion of the data structure is shown in figure 2. The highlighted path is for the first word in the bottom line of a five line address where the bottom line contains three words. The word class frequency table for this context is shown. It is seen that if an attention word occurs in this context it has between two and five characters or greater than or equal to nine characters. City words have at least three characters. City names with six characters are most common in this context with 77 occurrences. Conversely, if a two character word is detected at the indicated position, the frequency table says that it must be an attention word.

The frequencies stored in this data structure are derived from a large set of address block images. Each word was segmented and the line it occurred in as well as its position within the line, and its word class were recorded.

The confidence that a word class is correct is calculated by the following formula:

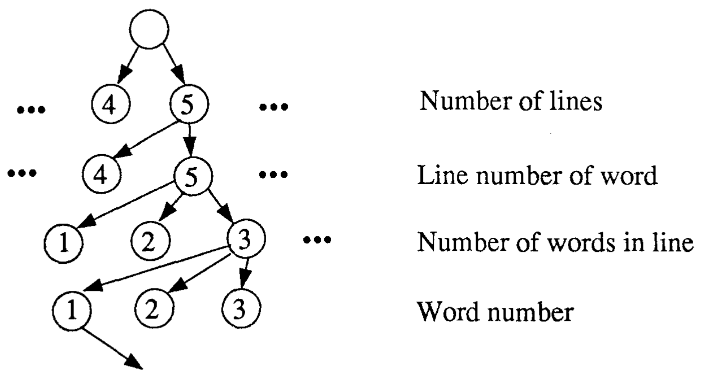
C(class | position, length, LD) =
f(class,position,length) / total (position,length)
** P(length | position, class) * P(LD | class)*

where,

f(class, position, length)
is the total number of occurrences of a given class in a given position with a given length. This is calculated by summing over a range of word lengths to compensate for errors in the length estimate.

total(position, length)
is the total number of words occurring at a given position, with a given length.

P(LD | class)
is a weight factor for whether the class is composed of letters or digits. This factor is set to 0.99 if the class matches the letter-digit decision and 0.01 if it does not. The non-zero value for not matching provides tolerance for errors in the letter-digit decision.



word	frequency at no. chars									
class	1	2	3	4	5	6	7	8	9	>=10
attn	0	2	2	19	6	0	0	0	5	12
city	0	0	4	10	45	77	20	19	18	19

Figure 2. Portion of data structure for word classes

P(length | position, class)
is a probability, that the word class at the given position has the length estimated from the image.

This last factor corrects errors that might be introduced by the compensation for imperfection in the word length estimate. For example, if the word in the image is "N." (a street prefix), then its estimated length would be 2. When the frequency of various types of words at this position is calculated, words of length 1, 2 and 3 are taken into the account. This is done to make the system less vulnerable to word length estimation errors. In this example, the extension of the word length range will cause all street names of length 3 to be counted. Since their frequency is higher than the frequency of street prefixes, the street name class might be chosen. The above factor can correct for this effect.

An example of calculating this confidence for the data shown in Figure 2 and the address image shown in Figure 3 is discussed below. Assume that a six character word composed of letters is detected as the first word in the last line of a five line address where the last line contains three words. Using a threshold of plus or minus one for the word length estimate,

C(attn | position, length, LD) =
6/142 * 0.99 * 6/46 = 0.005;
C(city | position, length, LD) =
136/142 * 0.99 * 136/212 = 0.608.

2.2. Horizontal Refinement

In this step, an ordered list of the configurations of word classes within each line are calculated. The objective is to use local, horizontal context within lines to refine the choices for each word. The inputs are the results of the initial assignment as well as a table of allowable configurations for lines and the a-priori probabilities of each configuration. The table is indexed by the number of lines in the address and the number of words in each line. The entries in the table are developed from analysis of a large set of training data.

The confidence of each horizontal configuration is calculated by multiplying its a-priori probability by the product of the confidences for each word. For example, if the bottom line of a five line address is considered, two horizontal configurations and their a-priori probabilities are:

city, state, ZIP (p = 0.89)
ATTENTION, name, name (p = 0.11)

If the results of the initial assignment were:

word 1: city (c = 0.6), ATTENTION (c = 0.3)
word 2: name (c = 0.4), state (c = 0.3)
word 3: ZIP (c = 0.8), name (c = 0.1)

then horizontal refinement would yield:

C(city, state, ZIP) = P(city, state, ZIP)
* C(city as word 1) * C(state as word 2)
* C(ZIP as word 3)
= 0.89 * 0.6 * 0.3 * 0.8
= 0.128

C(ATTENTION, name, name)
= P(ATTENTION, name, name)
* C(ATTENTION as word 1)
* C(name as word 2) * C(name as word 3)
= 0.11 * 0.3 * 0.4 * 0.1
= 0.001

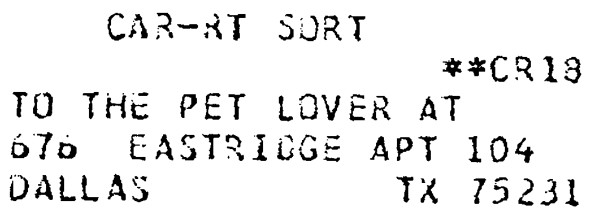

CAR-RT SORT
**CR13
TO THE PET LOVER AT
676 EASTRIDGE APT 104
DALLAS TX 75231

Figure 3. Example address block image

This has the effect of correcting the choice for the second word. Initially the best candidate was a personal name. After application of horizontal context, the correct choice of a state name was determined.

2.3. Vertical Refinement

This step uses the context *between* lines in an address block to further refine the classifications for each word. Input to this step are the results of horizontal refinement and a table of vertical configurations of line classifications and their a-priori probabilities. These probabilities are developed by analysis of a large set of training data. A line classification is a label assigned to a horizontal configuration of word classes. For example, a line that contains any street words is labeled as a street line. A line that contains a city, state and ZIP Code is labeled a city-state-ZIP line. Table 1 contains the line classifications used by the system.

Line Classifications	
city-state-zip	city-state
city-zip	state-zip
zip	
street	street-suite
suite-po-box	suite
po-box	building
rural	
attention	recipient
foreign	
po-box-city-state-zip	street-city-state-zip
other	

Table 1: Line Classifications Used by System

The purpose of this step is to use the global context of an address to refine the choices provided by local, horizontal context and thereby to improve the choices for each word.

The confidence of each vertical configuration of line classifications is calculated by multiplying its a-priori probability by the product of the confidences for each line. An example of this calculation is shown below. If the a-priori probabilities of two vertical configurations of three line addresses are:

P(NAME, STREET, CITY-STATE-ZIP) = 0.73
P(FIRM, CITY-STATE-ZIP, ATTENTION) = 0.27

and the confidences of the following horizontal configurations are input:

C(firm word, firm word, firm word as line 1) = 0.6
C(title, name, name as line 1) = 0.1

C(street number, street name, street suffix as line 2) = 0.7
C(city, state, ZIP as line 2) = 0.22

C(city, state, zip as line 3) = 0.128
C(ATTENTION, name, name as line 3) = 0.001

The confidences of the two vertical configurations are:

C(NAME, STREET, CITY-STATE-ZIP) =
P(NAME, STREET, CITY-STATE-ZIP)
* C(NAME as line 1) * C(STREET as line 2)
* C(CITY-STATE-ZIP as line 3)
= 0.73 * 0.1 * 0.7 * 0.128
= 0.007

C(FIRM, CITY-STATE-ZIP, ATTENTION) =
P(FIRM, CITY-STATE-ZIP, ATTENTION)
* C(FIRM as line 1)
* C(CITY-STATE-ZIP as line 2)
* C(ATTENTION as line 3)
= 0.27 * 0.6 * 0.22 * 0.001
= 0.00004

The final result of the parser is a ranked list of vertical configurations of line classifications and their confidences. A subset of the results of horizontal refinement that match each line classification and their confidences are also output.

For example, the highest ranked vertical configuration for a three line address may be NAME, STREET, CITY-STATE ZIP. Normally, there would be only one choice matching the CITY-STATE-ZIP line. However, for the street line, the following may be possible:

street number, street name, street suffix
street number, street name, apartment number
street number, street prefix, street name

This provides an ordered list of classifications for each word that has been determined by both horizontal and vertical context.

3. Experimental Results

A major objective of the parser is to locate and correctly classify the information needed to assign a nine digit ZIP Code to an address. This information includes the city name, state name and ZIP Code from which a 5-digit ZIP Code can be determined. The line in the address that contains the street address, P.O. Box rural route, or building name is even more important because it allows a 9-digit ZIP Code to be calculated for images that do not contain them. This line is generically termed the street address line (SAL). The performance of the parser was measured in locating the SAL, classifying it into one of the 4 classes mentioned above and classifying the individual words within the line.

The parser returns as its result a classification for each line. The SAL *location* is correct if the SAL found by the parser is in the same location as the actual SAL in the address image. The SAL *classification* is correct if the choice of SAL type (street, PO Box, building or rural route) by the parser is correct. The word classes within the SAL are correct if any of the word classes in the horizontal refinement results match the classes of the words in the image.

The error rate for locating and parsing SALs is the number of images in which the actual SAL was in a different line of the address block than returned by the parser. This is the most important measure of error since subsequent interpretation of the mailpiece depends on the location of the SAL. Some errors in parsing individual words are allowed.

The system was trained on images of 2450 machine-printed address blocks. Each block was segmented into isolated words and the text of each word and its word classification was typed into a file. The probabilities of word classes, line classifications, and vertical configurations mentioned earlier were all calculated from this data.

The system was tested on 729 images of address blocks. The lines and words were segmented from these images and the number of characters in each word were estimated by a program that provided a range of possible lengths (e.g., a word might contain 3 or 4 characters). An estimation of whether the words contained letters or digits was also used. The results of testing are shown in Table 4. Overall, the parser was 94 percent correct at locating the SAL and 91 percent correct at classifying it. The classifications for all the words within every SAL were correct in 88 percent of the cases.

measurement	number correct	percent correct
total	729	100%
locating SAL	685	94%
classifying SAL	663	91%
word classes of SAL	642	88%
rejects	29	4%
errors in locating SAL	15	2%

Table 4. Performance in locating and parsing street address line

The performance of the parser in assigning the correct categories to every line and every word within a line is shown in Tables 5 and 6. Table 5 is for a subset of the test images that were assigned a ZIP Code by a commercial address reader and Table 6 is for a subset of the test images that were rejected by the same commercial reader. Rejects are often caused by poor quality images that contain many broken or touching characters. Each row of the tables refer to specific lines in an

line	total images	reject	vertical choice									
			1					2				
			class	word choice				class	word choice			
				1	2	3	4		1	2	3	4
bottom	473	2%	96%	79%	95%	95%	95%	96%	96%	96%	96%	96%
second from bottom	473	2%	90%	69%	81%	84%	85%	93%	88%	89%	89%	89%
third from bottom	457	2%	90%	62%	76%	80%	82%	92%	86%	86%	86%	86%
fourth from bottom	104	4%	85%	51%	68%	75%	75%	91%	81%	84%	84%	84%
fifth from bottom	76	11%	74%	57%	63%	65%	66%	81%	76%	78%	78%	78%

Table 2. Performance in classifying individual lines and words (accepted by commercial address reader)

line	total images	reject	vertical choice									
			1					2				
			class	word choice				class	word choice			
				1	2	3	4		1	2	3	4
bottom	216	6%	90%	79%	89%	90%	90%	90%	90%	90%	90%	90%
second from bottom	214	6%	83%	66%	76%	76%	78%	84%	81%	81%	81%	81%
third from bottom	210	6%	87%	52%	70%	75%	76%	88%	82%	82%	82%	82%
fourth from bottom	100	11%	80%	54%	66%	69%	71%	86%	78%	78%	78%	78%
fifth from bottom	19	32%	58%	32%	53%	53%	58%	68%	63%	68%	68%	68%

Table 3. Performance in classifying individual lines and words (rejected by commercial address reader)

address. The number of images in each line is shown as well as the percentage of lines that were rejected. The performance of the top two vertical choices is also given. For each vertical choice, the performance in assigning the correct class to each line is shown in the column labeled "class". The performance in assigning the correct class to every word within a line is shown in the columns labeled "word choice." The performance of the top four choices is given. A configuration of word choices was considered correct if all word classifications chosen by the system were identical to the actual ones. It is seen that the performance in line classification ranges from 96 percent correct for the bottom line to 74 percent for the fifth line from the bottom, in images that were sorted by the commercial reader. The performance in assigning classes to each word in these lines ranges from 95 percent correct for the bottom line to 66 percent for the fifth line, if the top four choices are considered.

The performance of the system was reported separately for lines 1 through 5 because of the diversity of word classifications that are output. Also, the structure of lines contain varying amounts of information about the classifications of the words. The information content is similar in the bottom lines, while it gets more diverse and less structured in the upper lines. This also explains the better performance of the system for lower lines. The poorer performance for upper lines should not be seen as a disadvantage, since those lines usually do not contain information needed to assign a ZIP+4 code to the address.

4. Summary and Conclusions

A pattern matching algorithm for parsing postal addresses was presented. Such algorithms have been very successful at parsing natural language from constrained domains. An objective of the work discussed here was to determine if such a technique would work well for *images* from a constrained domain. It was seen that the proposed algorithm overcame the limitations of other grammatical approaches (such as ATNs) that require near perfect recognition and exhaustive dictionaries for every word by not using recognition results and instead relying on only structural information. This information included the number of lines in an address, the number of words in each line, the number of characters in each word, and an estimate of whether each word contained letters or digits.

The testing and further development of the system is continuing. The usefulness of recognition information is being investigated. A limited amount of word recognition will be performed in selected situations. An example of how this could improve performance is if the first word in the last line of an address were determined to be either ATTENTION or a city name. If it is recognized as "ATTN", the probability of this class can be set very high and the probability of this word being a city name can be set to almost 0. It is expected that such an integration of recognition information will improve performance.

Acknowledgments

The support of the Office of Advanced Technology of the United States Postal Service is gratefully acknowledged. Carl O'Connor of USPS as well as Gerardo Garcia and Gilles Houle of Arthur D. Little, Inc. provided helpful encouragement. Electrocom Automation and Piotr Prussak of SUNY at Buffalo produced the data files on which the system was trained. Ed Cohen provided helpful comments on the manuscript.

References

1. T. A. Bayer, "Interpretation of structured documents in a frame system", *Pre-Proceedings of SSPR90 the IAPR Workshop on Syntactic and Structual Pattern Recognition*, Murray Hill, N.J., June 13-15, 1990, 47-56.
2. J. G. Carbonell and P. J. Hayes, "Recovery strategies for parsing extragrammatical language", *The American Journal of Computational Linguistics 9*, 3-4 (July-December, 1983), 123-146.
3. G. DeJong, "Prediction and substantiation: a new approach to natural language parsing", *Cognitive Science 3* (September 1979), 251-273.
4. J. L. Fisher, S. C. Hinds and D. P. D'Amato, "A rule-based system for document image segmentation", *10th International Conference on Pattern Recognition*, Atlantic City, June 16-21, 1990, 567-572.
5. A. Hanson and E. Riseman, "Segmentation of natural scenes", in *Computer Vision Systems*, A. Hanson and E. Riseman (editor), Academic Press, 1978, 129-163.
6. J. J. Hull, "Character recognition: the reading of text by computer", in *The Encyclopedia of Artificial Intelligence*, S. C. Shapiro (editor), John Wiley and Sons, 1987, 82-88.
7. P. W. Palumbo, "Two-dimensional heuristic augmented transition network parsing", *The Second Conference on Artificial Intelligence Applications*, Miami Beach, Florida, December 11-13, 1985, 396-401.
8. R. C. Parkinson, K. M. Colby and W. S. Faught, "Conversational language comprehension using integrated pattern-matching and parsing", *Artificial Intelligence 9* (1977), 111-134..
9. S. C. Shapiro, "Generalized augmented transition network grammars for generation from semantic networks", *The American Journal of Computational Linguistics 8*, 1 (January-March, 1982), 12-25.
10. J. Weizenbaum, *Computer power and human reason: from judgement to calculation*, W.H. Freeman, San Francisco, 1976.
11. T. Winograd, *Language as a cognitive process*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.