Integrating Geometrical and Linguistic Analysis for Email Signature Block Parsing

HAO CHEN
University of California at Berkeley
JIANYING HU
Lucent Technologies Bell Labs
and
RICHARD W. SPROAT
AT&T Labs—Research

The signature block is a common structured component found in email messages. Accurate identification and analysis of signature blocks is important in many multimedia messaging and information retrieval applications such as email text-to-speech rendering, automatic construction of personal address databases, and interactive message retrieval. It is also a very challenging task, because signature blocks often appear in complex two-dimensional layouts which are guided only by loose conventions. Traditional text analysis methods designed to deal with sequential text cannot handle two-dimensional structures, while the highly unconstrained nature of signature blocks makes the application of two-dimensional grammars very difficult. In this article, we describe an algorithm for signature block analysis which combines two-dimensional structural segmentation with one-dimensional grammatical constraints. The information obtained from both layout and linguistic analysis is integrated in the form of weighted finite-state transducers. The algorithm is currently implemented as a component in a preprocessing system for email text-to-speech rendering.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Selection process*; H.4.3 [**Information Systems Applications**]: Communications Applications—*Electronic mail*

General Terms: Algorithms

Additional Key Words and Phrases: Email signature block, finite-state transducer, geometrical analysis, linguistic analysis, text-to-speech rendering

Authors' addresses: H. Chen, School of Information Management and Systems, University of California at Berkeley, Berkeley, CA 94720-4600; email: hchen@sims.berkeley.edu; J. Hu, Lucent Technologies Bell Labs, 700 Mountain Avenue, Murray Hill, NJ 07974-0636; email: jianhu@bell-labs.com; R. W. Sproat, AT&T Labs—Research, 180 Park Avenue, P.O. Box 971, Florham Park, NJ 07932; email: rws@research.att.com.

```
_    /| Vinod Anupam                email: anupam@research.bell-labs.com
 'O.o'  Bell Labs, Lucent Tech.     www: http://www.tempo.lucent.com/~anupam
=(___)= 700 Mountain Ave., Rm 2C-236A phone: (908)582-7366
   U    Murray Hill, NJ 07974-0636   fax: (908)582-5809
```

Fig. 1. A signature block.

1. INTRODUCTION

The rapidly increasing use of the Internet in recent years has made email one of the most common forms of business and personal communication. How to manage the large and dynamic collections of email documents for efficient storage and information retrieval and how to provide conversions between email and other forms of messages (e.g., voice mail and fax) to allow convenient access whenever and wherever the user needs are some of the most important research areas in multimedia messaging.

The content of modern-day email has expanded beyond text to include encoded documents, images, even audio and video clips. However, unmarked text is still the prevailing format for communication with email, due to its simplicity and sufficiency in terms of conveying ideas, conducting discussions, making announcements, etc. One of the most common structured elements in text email is the signature block. It contains information about the sender, such as email address, Web address, telephone and fax numbers, personal name, postal address, etc., and is usually separated from the rest of the message by some sort of border. Accurate identification and parsing of signature blocks is important for many multimedia messaging applications such as email text-to-speech (TTS) rendering, automatic construction of personal address databases, and interactive message retrieval.

However, parsing of signature blocks is also a very challenging task due to the fact that they often appear in complex two-dimensional layouts which are guided only by loose conventions. Figure 1 shows one example of such layouts. Apparently a straightforward line-by-line analysis using conventional text analysis methods will fail to extract fields such as the postal address. The only way to extract functional fields from such layouts is to combine two-dimensional layout analysis with linguistic constraints.

In this article, we describe a new approach to combining two-dimensional structural analysis with one-dimensional grammatical constraints for signature block parsing. The information obtained from both layout and linguistic analysis is integrated in the form of weighted finite-state transducers (WFST) [Mohri et al. 1997; Pereira and Riley 1996], and the final solution is the optimal interpretation under both constraints. We will focus on the parsing of the signature block, assuming it is already identified, and will briefly explain how a different version of the algorithm can be used to improve the identification of signature blocks as well.

The algorithm is currently implemented as a component of an email text-to-speech rendering system called *Emu* [Sproat et al. 1998]. Automatic conversion of email into speech is one of the most important commercial

```
#\     Andy Elms     -- Tel: +44 1483 300800 x2753 -- Fax: +44 1483 34139    /#
##\   Dept. of Elec. Eng., University of Surrey, GUILDFORD, GU2 5XH, UK.    /##
###\ A.Elms@ee.surrey.ac.uk -- http://www.ee.surrey.ac.uk/Personal/A.Elms  /###
```

Fig. 2. One-column layout of a signature block.

applications of text-to-speech technology, and is one technological component of the growing interest in *media conversion*.

In the Emu system, an email message is first parsed into different regions (headers, quoted material, and signature blocks, among others), and these regions are marked with tags that indicate the regions' properties: the algorithm described in the current article is used in this first phase to identify signature blocks, and to parse them into meaningful components. Second, a normalization of the text is computed. The normalization performed in this second phase largely involves the expansion of unusual "words" (*WinNT*), as well as email addresses, URLs and other nonstandard material. The output of the normalization phase is "device independent" in the sense that the normalizations performed produce text that is appropriate as input to any (English) TTS system. Finally, in the third phase, the marked-up and normalized text is *rendered* by converting it into text interspersed with control sequences for the Bell Labs American English TTS system [Sproat 1997].

2. RELATED WORK

Document layout segmentation and logical structure analysis have been studied by many researchers in the context of understanding of printed documents, including journal pages, newspaper articles, business letters, mail pieces, forms, catalogs, etc. While in some sense email text can be viewed as a special form of printed document, there are also important differences. On the one hand, unlike printed documents, the text content in the case of email is readily available and free of noise. On the other hand, since email messages are not formal publications, there are few rules regarding the layout structure of signature blocks, as demonstrated by examples shown in Figures 2, 3, 4, and 5. This higher degree of variability makes layout segmentation a more challenging task.

Many different approaches have been developed over the years for printed document layout segmentation, which can be roughly defined as the segmentation of a document page into blocks of more or less coherent content. The most notable ones include the recursive projection profile cuts method [Mizuno et al. 1991; Nagy and Stoddard 1985; Wang and Srihari 1989], the approach based on maximal white rectangles [Baird 1992; Baird et al. 1990], and some other methods based on the analysis of background white spaces [Antonacopoulos and Ritchings 1994; Pavlidis 1991; Rahgozar et al. 1994; Rus and Summers 1994]. Each of these techniques relies, to a different extent, on assumptions about the generic document layout structure, particularly, rectangularity of text blocks and white spacing around each block. Unfortunately such assumptions do not always hold in the case

```
Bob Carpenter                    Email: carp@research.bell-labs.com
Lucent Technologies Bell Labs    WWW: http://macduff.andrew.cmu.edu/carpenter
600 Mountain Avenue, 2D-329      Voice: 908 582-5790
Murray Hill, NJ  07974           Fax: 908 582-3306
```

Fig. 3. Two-column layout of a signature block.

```
-------------------- mailto:lyon@research.apple.com --------------------
  Dick Lyon                                        Distinguished Scientist
  Apple Computer 301-3M                                Apple Research Labs
  One Infinite Loop                                  phone: (408) 974-4245
  Cupertino  CA 95014                                  fax: (408) 974-8414
-------------- http://www.research.apple.com/personal/lyon/ -------------
```

Fig. 4. Variable number of columns layout of a signature block.

```
+-----------------------------------------------------------------+
|  Stefan Reichling                                               |
|                                Anorganisch-Chemisches Institut |
|  Tel +49-6221-548649              Universitaet Heidelberg      |
|  Fax +49-6221-545707                  69120 Heidelberg         |
|  e-mail: steve@indi.aci.uni-heidelberg.de      Germany         |
|  www: http://www.rzuser.uni-heidelberg.de/~il1                 |
+-----------------------------------------------------------------+
```

Fig. 5. Nonrectangular columns in a signature block.

of email signature blocks. For example, Figure 5 shows an example containing nonrectangular blocks which cannot be separated by a vertical cut, and Figure 4 shows an example where different layout structures (one-column and two-column) are placed directly on top of each other with no white space in between. We introduce a new approach based on recursive foreground-background connected-component analysis to handle such "unconventional" layout structures encountered in email signature blocks.

Fewer studies have been carried out on logical layout analysis, which involves functional labeling of document blocks. Most of the previous approaches rely on geometric features alone. Some researchers have used texture analysis or other visual features such as font size, location and aspect ratio of the block, indentation attributes of the block, etc. to distinguish text blocks from images and graphics, or to assign high-level labels to text blocks such as titles, captions, paragraphs, itemized lists, tables, etc. [Etemad et al. 1997; Jain and Bhattacharjee 1992; Rus and Summers 1994; Wang and Srihari 1989]. The features used in these approaches do not always translate to email documents. Furthermore, finer logical labels are not obtained by such analysis. In Porter and Rainero [1992], more details of logical layout structure are recovered using labels provided in a particular formatting language (e.g., Latex or PostScript). The method does not apply to generic, unmarked documents. Other researchers have applied more detailed domain knowledge in the forms of block grammars [Nagy et al. 1992], array grammars [Takasu et al. 1993; 1994], geometric trees [Dengel and Barth 1988], or specialized tools [Srihari et al. 1987] to obtain finer-level logical labels in specific document forms such as business letters, pages from a particular journal, and postal pieces, based on strict layout rules. These techniques cannot be applied to email signature block analysis, where the layout design is highly unconstrained and where geometric attributes alone are not sufficient to distinguish between different functional entities (e.g., postal address versus telephone numbers). In our system, loose geometric layout conventions are integrated with linguistic analysis to achieve reliable logical labeling of all major functional classes encountered in email signature blocks.

3. PROBLEM DEFINITION

A signature block usually appears at the end of a message, although it may also be in the middle of a message if there is a postscript or quoted message. It is used to indicate contact information, such as an email address, Web address, telephone and fax numbers, name, postal address, and even quotes and other miscellaneous text. Unlike other parts of the email message, the signature block is highly unconstrained in that it is quite personalized and in that there are hardly any style restrictions. We are faced with both geometrical complexity and linguistic complexity when analyzing the signature block.

—*Geometrical Complexity:* Geometrical properties indicate the reading sequence in a signature block. The simplest layout of a signature block has only one column, and we read from top to bottom, and left to right on each line (Figure 2). However, for esthetic reasons, and to shrink the length, signature blocks often have rather complicated layouts, as discussed in the previous section. These various styles complicate the analysis task.

—*Linguistic Complexity:* Some components of the signature block, such as email and Web addresses, have strict patterns and are easily recognizable. Others, such as personal names and postal addresses, have few lexical constraints. Worse yet, occasionally there are quotes or other miscellaneous text in the signature blocks, which have no lexical constraints at all. Humans identify these components by the semantics of natural language. However, natural language understanding by computer is as yet an unsolved problem.

Before further discussion, we define some terms used in the following analysis. A *signature block*, as shown in the previous examples, is part of an email message. It is composed of several continuous lines of text which are used primarily to indicate personal contact information. A signature block may be decomposed into *reading blocks*. Text in a reading block can be read out in a meaningful order by simply following the sequence from top to bottom, and from left to right on each line. Text in one reading block is normally read out completely before going into another reading block. A reading block is decomposed further into *functional blocks*. Text in each functional block belongs to the same *functional class*. Ten functional classes are defined in the current system: (1) email address, (2) Web

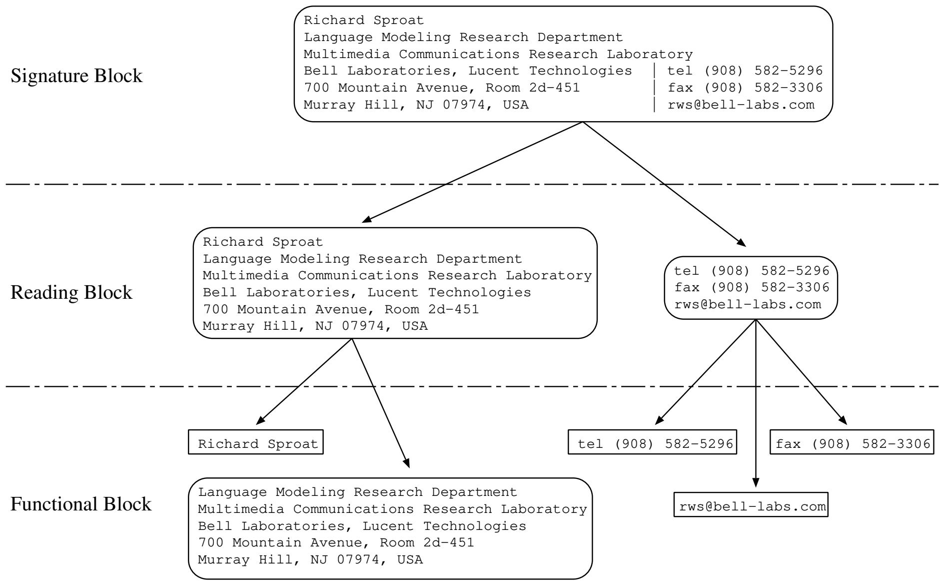


Fig. 6. Hierarchical text structure.

address, (3) telephone number, (4) fax number, (5) personal name, (6) postal address, (7) title, (8) quote, (9) stub (auxiliary words, such as "`home`" or "`office`" after a telephone number), and (10) miscellaneous text. Any text that is not related to any of the first nine functional classes is miscellaneous text. Signature blocks, reading blocks, and functional blocks constitute a hierarchical text structure, as shown in Figure 6.

The flowchart of signature block analysis is shown in Figure 7. The input to the system is a signature block extracted from a message with `TAB` characters expanded to space characters. Then, geometrical analysis and linguistic analysis are applied. Finally, the signature block is broken down to several functional blocks, each related to a functional class.

4. GEOMETRICAL ANALYSIS

Geometrical analysis breaks a signature block down to one or more reading blocks, where text in each reading block can be read out continuously. Text in a reading block is usually grouped together, which is easily identified using connected-component analysis (Section 4.2). However, there are cases where different reading blocks are connected (Figure 4) and where the standard connected-component analysis technique needs to be modified (Section 4.3).

There are several algorithms for connected-component analysis, and we choose the Line Adjacency Graph (LAG) algorithm [Pavlidis 1982]. This is a bottom-up approach, where each line in the text region is broken into several line segments. Overlapping line segments on adjacent lines are placed into the same connected component, and all line segments in a connected component are found from the transitive closure.

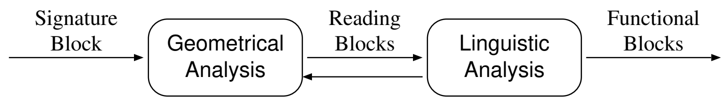


Fig. 7. Flowchart of signature block analysis.

4.1 Line Segment Extraction

A line is first broken into line segments, which are the smallest units for functional class labeling. It is important to use line segments instead of lines because in some layout styles (e.g., Figure 2) a single line contains multiple line segments, each belonging to a different functional class. Obviously, characters in the same line segment should belong to the same reading block, but different line segments may or may not be in the same reading block. Intuitively, characters that are close to each other should be assigned to the same line segment, whereas visually separated characters should be assigned to different line segments. The assignment is performed using *disconnectedness scores* and threshold values on nonalphanumerics. Those which visually indicate segmentation points, such as "|" and ",", are assigned high positive disconnectedness scores, while those which visually indicate connection points, such as ":" and "–", are assigned high negative disconnectedness scores. Each string of nonalphanumerics whose sum of disconnectedness scores is greater than the threshold is considered as a separator, which assigns its surrounding text into two different line segments.

The disconnectedness scores and threshold values are estimated empirically. Apparently, certain segmentation ambiguities cannot be resolved completely using geometrical information alone. They will be further analyzed with linguistic information taken into account in the linguistic analysis stage as discussed in Sections 5.2.2 and 5.3.1.

4.2 Connected-Component Analysis

Line segment extraction horizontally connects closely related individual characters into line segments. The next step is to extract vertically connected line segments.

In the traditional LAG algorithm [Pavlidis 1982], two line segments on adjacent lines are considered *vertically connected* if they overlap, i.e., they have at least one x-coordinate in common. However, this simple rule causes some problems in the signature block analysis. In Figure 4, although the line segment on the first line overlaps with each of the line segments on the second line, they actually belong to different reading blocks. For human vision, two vertically adjacent line segments must overlap *considerably* to have the effect of being visually connected, which is reflected in our definition of *vertical connectedness*.

Two line segments $L_1((x_A, y), (x_B, y))$ and $L_2((x_C, y + 1), (x_D, y + 1))$ are considered *vertically connected* if and only if

(1) $x_A < x_D$ and $x_B > x_C$ (i.e., L_1 and L_2 overlap) and

```
-- "A friend in need is a friend in deed." --
Nematollaah Shiri            Office: LB 1041-1
Concordia University         Tel: (514) 848-3033
1455 de Maisonneuve West     Fax: (514) 848-2830
Montreal, Quebec, H3G 1M8    shiri@cs.concordia.ca
URL http://www.cs.concordia.ca/~grad/shiri/
```

Fig. 8. Mixed reading blocks.

(2) $\min(x_B - x_C, x_D - x_A)/\min(x_B - x_A, x_D - x_C) >$ threshold.

The transitive closure of all pairs of vertically connected line segments defines a connected component. There are many existing algorithms for efficiently computing the transitive closure. We choose the algorithm for computing equivalence classes as described in Horowitz et al. [1993]. It has the time complexity of $O(M + N)$, where M is the number of line segments, and N is the number of pairs of vertically connected line segments.

4.3 Mixed Reading Blocks

Usually a connected component contains only one reading block. However, there are a few cases where there is more than one reading block in a connected component. Figure 8 is a typical example where several reading blocks are juxtaposed in the middle and where the reading block at the top or bottom connects them together. The top or bottom reading block is so long that the principle of *vertical connectedness* does not help to break them from the middle ones.

To detect the mixed reading block, line segment extraction and connected-component analysis are performed on all *background* characters. Background characters are space characters, and a background connected component is comprised of connected space characters. A background connected component is a *separator* if (1) at least one line segment of the background connected component is in the middle of the reading block; in other words, it does not touch the left or right margin of the reading block; and (2) the total height of the background connected component is greater than a threshold. Figure 9 shows a case where the background connected component is a separator. (The background connected component is filled with "#")

If a separator is found, the corresponding reading block is broken into three new blocks. The first one contains line segments which are above the separator. The second one contains line segments which are below the separator. The third one contains the remaining line segments from the old reading block. In fact, the first and second new reading blocks are the top and bottom block in the old reading block, respectively, and the third one contains all the juxtaposed blocks in the middle of the old block. After that, each new reading block goes through the ordinary connected-component analysis again. Figure 10 shows the procedure and the result.

```
-- "A friend in need is a friend in deed." --
Nematollaah Shiri###########Office: LB 1041-1
Concordia University#########Tel: (514) 848-3033
1455 de Maisonneuve West#####Fax: (514) 848-2830
Montreal, Quebec, H3G 1M8####shiri@cs.concordia.ca
URL http://www.cs.concordia.ca/~grad/shiri/
```

Fig. 9. Background connected component which is a separator.

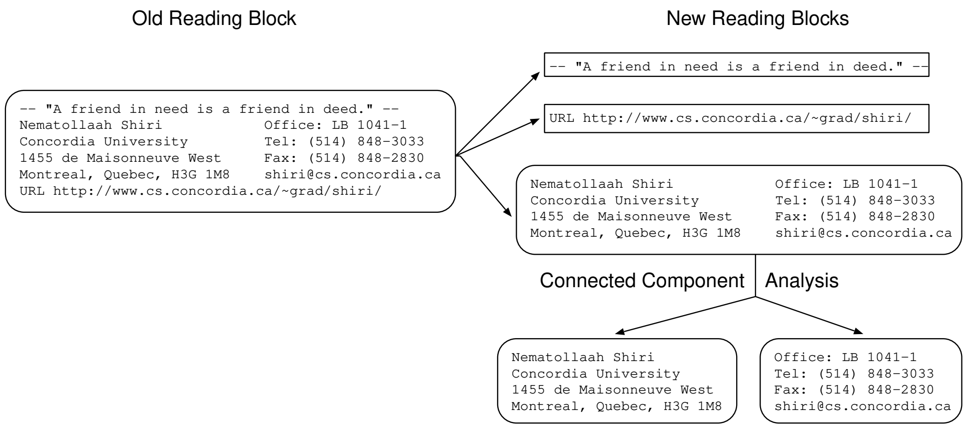


Fig. 10. Breaking mixed reading blocks.

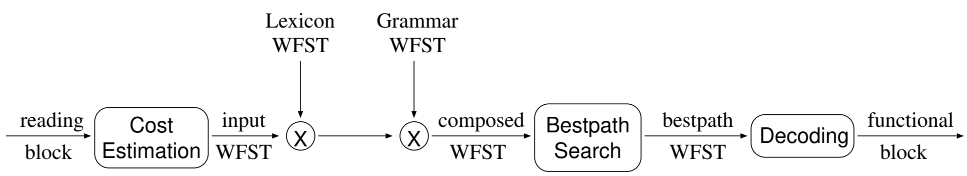


Fig. 11. Linguistic analysis.

4.4 Remaining Errors

Due to the empirical selection of disconnectedness scores and the limitation of connected-component analysis to resolve geometrical ambiguity, both undersegmentation and oversegmentation errors could result from the geometrical analysis. In fact, most of them are not remediable unless lexical knowledge is applied. While the next stage, linguistic analysis, serves primarily to detect functional classes, it will also correct most of the remaining segmentation errors by combining geometrical analysis with linguistic constraints.

5. LINGUISTIC ANALYSIS

Linguistic analysis breaks a reading block into several functional blocks and relates each functional block with a functional class. Linguistic analysis is carried out using weighted finite-state transducers (WFST) [Mohri et al. 1997; Pereira and Riley 1996] as shown in Figure 11. First, the cost of relating a line segment with each functional class is estimated. Then, an input WFST is built, which incorporates all possible choices with their costs. The input WFST is then composed with a lexicon WFST describing the construction of a functional block from line segments, and a grammar WFST describing the construction of a reading block from functional blocks. Finally, the functional class of each line segment is revealed from the optimal path in the composed WFST. Most of the oversegmentation and undersegmentation errors resulting from geometrical analysis are also corrected in linguistic analysis.

In the following we give a brief and informal description of weighted finite-state transducers and the relevant operations. The formal definition as well as detailed discussions on their properties and operations can be found in Pereira and Riley [1996] and Mohri et al. [1997].

5.1 Weighted Finite-State Transducers

A weighted finite-state transducer contains a set of *states* with a distinguished *start state* and one or more *final states*. Each state except the final state has a number of *arcs* to other states. Each arc has an *input symbol*, an *output symbol*, and a *cost*. Figure 18 shows a WFST. A finite-state acceptor (FSA) (Figure 14) can be thought of as a particular case of WFST, where the input symbol is identical to the output symbol on each arc, and where the cost on each arc is the *free* cost (usually 0).

Following any path leading from the start state to the final state in an WFST, there are an *input string* (string of input symbols), an *output string* (string of output symbols), and a *total cost* (the sum of all costs on the path). The WFST is said to *transduce* the input string into the output string with the total cost.

The *composition* of two WFSTs is a new WFST such that if the first WFST transduces string $s1$ into $s2$ with cost $c1$ and the second WFST transduces string $s2$ into $s3$ with cost $c2$, the new WFST transduces $s1$ into $s3$ with cost $c1 + c2$.

The *best-path* algorithm searches an WFST for the optimal path leading from the start state to the final state in the sense that it has the minimum total cost. This best path is represented as a single-path WFST.

WFSTs have been widely used in natural language processing [Sproat 1996]. They have also been shown to be powerful techniques for speech and handwriting recognition, where the recognition process is viewed as a cascade of weighted finite-state transductions from the input signal sequence to a word or sentence in a given language [Guyon et al. 1996]. In our current problem, the process of linguistic analysis is formalized as a cascade of transductions from line segments to functional blocks.

5.2 Cost Estimation

For each line segment in the reading block, there are a pair of neighboring nodes in the input WFST connected by several arcs. On each arc, the input/output symbol represents a functional class, and the cost reflects how

Table I. Functional Classes

Symbol	Functional Class	Example
E	Email address	`jws@research.bell-labs.com`
W	Web address	`http://www.bell-labs.com/who/jws`
P	Phone number	`(908)582-3433`
F	Fax number	`(908)582-7308`
N	Personal name	`John W. Smith`
A	Postal address	`700 Mountain Avenue, Murray Hill, NJ 07974`
T	Title	`Associate Professor`
Q	Quote	`"640K ought to be enough for everyone"`
S	Stub	`home` (e.g., following a phone number)
M	Miscellaneous text	`Address valid until Aug 29, 1997`

likely the line segment is related to that functional class. We have identified nine major functional classes, plus a 10th miscellaneous text class (Table I). In addition, two more symbols are used to represent the line break (`L`) and boundary between reading blocks (`B`).

Text relating to the first four functional classes (email address, Web address, telephone and fax numbers) has a relatively strict pattern. These classes are termed *strict classes*. The remaining six classes (personal name, postal address, title, quote, stub, and miscellaneous text) are termed *loose classes*, since they have rather free styles. Cost estimation is quite different between strict classes and loose classes.

5.2.1 *Cost Estimation for Strict Classes*. Text belonging to strict classes is identified by regular-expression matching, using a finite-state linguistic analysis toolkit, described briefly in Sproat [1996]. Care is taken to account for different writing styles of email address, Web address, and telephone and fax numbers. If the entire text in a line segment matches the regular expression, the corresponding functional class is assigned a low cost, and all other classes are assigned higher costs.

5.2.2 *Correction of Undersegmentation Errors*. Many undersegmentation errors resulting from geometrical analysis for line segment extraction can be detected during the cost estimation for strict classes. Figure 12 shows a typical undersegmentation error where the email address is placed in the same line segment as the telephone number because they are so close to each other. This kind of error cannot be detected by geometrical analysis alone. To detect it, after successfully matching the entire text against the regular expression for telephone number, the matched telephone number as well as keywords indicating a telephone number (such as `tel`, `phone`, `voice`) are removed from the original text. The remaining text is checked for any alphanumerics. If such are found, it indicates that the line segment contains other text, which signals an undersegmentation error. Then, resegmentation is performed on the line segment by breaking it at each word boundary. This seems to lead to oversegmentation very easily, but that problem will be taken care of by the language-directed segmentation algorithm to be discussed in Section 5.3.1.

```
Tel:  908 582 1211 E-mail:  Koen@research.bell-labs.com
```

Fig. 12. Undersegmentation error (the bounding box is not part of the text).

5.2.3 *Cost Estimation for Loose Classes*. Since there are no strict patterns for text relating to loose classes, it is mostly identified by some commonly observed conventions. For example, the first letter of each word is usually capitalized in names and addresses, but not in quotes or miscellaneous text; quotes are usually contained in quotation marks; digits tend to appear more frequently in addresses than other classes. Contrary to strict classes where the estimated costs are either low (for likely classes) or high (for unlikely classes), the confidence in identifying loose class text is much lower, and the estimated costs among different functional classes do not differ as much.

Cost estimation for loose text classes is not highly reliable due to their vague patterns. This especially causes trouble in distinguishing personal names from city names, since there are very few rules guiding the composition of personal names, and in fact many personal names are easily confused with city names. Although the `From:` field of a message sometimes contains the personal name of the sender, it often does not. In one rough estimate that we carried out, out of 1985 email messages collected in a company internal mailbox, 287 (14%) do not contain personal names in their `From:` fields. It is expected that this percentage would be higher for messages from a more widely distributed set of sources. Naturally, when the personal name does occur in the `From:` field we can use that information to detect the personal name in the signature block; but when it is absent we must resort to other methods. In general, one might assume that identifying a personal name is simply a matter of dictionary lookup. Unfortunately this is not sufficient. It is simply unrealistic to assume one can have a complete dictionary of personal names. Consider that there are over a million distinct (family) names in the United States alone; and when one considers not just email from United States residents, but from the entire world, the number of distinct names becomes much larger. The problem for constructing a dictionary of names is not so much one of storage as of *acquisition*: it is impractical to store beforehand all the names that one might encounter. We have therefore explored an alternative approach, which uses the sender's user name as a clue to finding the personal name in the signature block. We discuss this approach in the next section.

5.2.4 *Personal Name Identification*. More often than not, the email user name is derived from the real personal name. The derivation often observes the following rules:

—A user name is constructed by concatenating letter strings directly with possible additional punctuation characters.

—The letter strings must be prefixes of the first name, middle name, or family name.

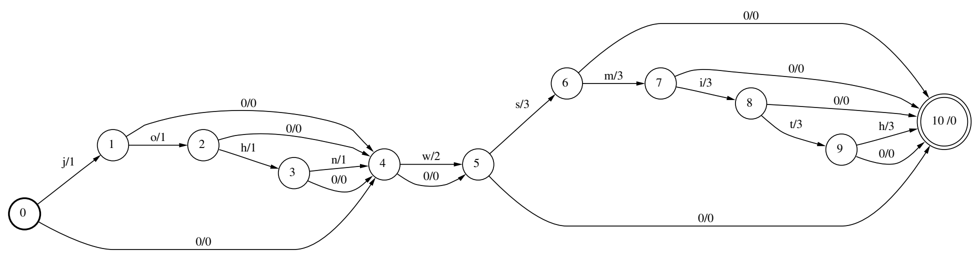


Fig. 13. Well-formed user name FST for "John W. Smith". Here, and elsewhere, "0" after "/" on an arc represents a cost (the *free* cost), and any other "0" on an arc represents the null string ϵ (which allows one to transition to the next state without consuming any input). Thus "0/0" represents an ϵ-labeled arc with a free transition cost.

—Each of the first name, middle name, or family name may contribute zero or one prefix as a substring of the user name.

User names constructed by these rules are termed *well-formed* user names. For example, from personal name "`John W. Smith`", "`jws`", "`jwsmith`", "`johnsmith`", "`johns`", "`smith`" are all well-formed user names, whereas "`s_jws`" is not.

It is easy to automatically construct a finite-state transducer (FST) which enumerates all possible well-formed user names from a given personal name. Figure 13 shows such an FST for personal name "`John W. Smith`". The input symbol on each arc is a letter from the personal name, and the sequence of input symbols on any path leading from the source node to the destination node constitutes a well-formed user name. The output symbol on an arc indicates which part of the full name (first name, middle name, or family name) this arc corresponds to. It is used to quantitatively estimate how likely it is that an unknown phrase is a personal name, as discussed later.

To estimate if a candidate phrase is a personal name, a well-formed user name FST is constructed from the candidate phrase, assuming that it is a personal name. Then, a single-path FSA which generates the user name is constructed (Figure 14 shows such an FSA for user name "`jws`"). The single-path FSA is then composed with the well-formed user name FST, and a best-path search is performed. If the best-path FST is nonempty, it indicates that the phrase is likely to be a personal name, and thus a low cost is assigned for relating the phrase to the personal name functional class.

It is not sufficient to only qualitatively identify a personal name. Due to segmentation errors from the geometrical analysis stage as well as the requirements from the lexicon-directed segmentation algorithm (Section 5.3.1), it is imperative to quantitatively estimate how likely a candidate phrase is to be a personal name. For example, to resolve the segmentation ambiguity on a line which reads "`John W. Smith Chairman`", the algorithm should be able to tell that "`John W. Smith`" looks more like a personal name than "`John W. Smith Chairman`" (without a knowledge of semantics).

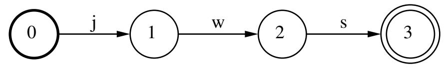


Fig. 14. User Name FSA for "jws".

This is achieved by making use of the output symbols in the well-formed user name FST. The output symbol indicates which part of the personal name a letter in the user name comes from. By counting the number of different output symbols in the best-path FST, it is revealed how many parts of the personal name contribute to the user name. The more parts that do not contribute to the user name, the less likely that the candidate phrase is a personal name. For example, each part of "`John W. Smith`" contributes to the user name "`jws`"; but the last part of "`John W. Smith Chairman`" does not, so "`John W. Smith`" is assigned a lower cost *qua* personal name than "`John W. Smith Chairman`".

Personal name identification is complicated when the name is comprised of a one-word first name and a one-word family name. This could belong to one of the following three common cases:

—The person does not have a middle initial.

—The person has a middle initial, but it is omitted in this instance.

—The person does not have a middle initial. However, the first name actually represents two Asian, e.g., Chinese, characters.

To deal with the second case where the middle initial is omitted (and therefore unknown) from the written personal name, all 26 letters are considered as candidates for the middle initial. Figure 15 shows the well-formed user name FST from personal name "`John Smith`" where the middle initial "`W`" is omitted. Note, that, since the user name is case insensitive, all letters in the personal name are changed to lowercase. All punctuation symbols in the user name, if any, are removed before the user name is matched against the FST.

In the third scenario, the first name is actually composed of two segments, each of which represents an Asian character. Since the boundary of the two segments is unknown, all possible boundary positions are presented to the well-formed user name FST.

Based on the concept of well-formed user name and through the use of FST, personal names, which belong to the loose functional class, can be identified with much higher confidence.

5.3 The Input WFST

An input WFST is built for each reading block. For each line segment in the reading block, there are a pair of neighboring nodes in the WFST connected by several arcs. The input/output symbol of the arc represents a functional class, and the cost indicates the likelihood that the line segment is related to that functional class. Figure 16 shows a reading block and its input WFST. (Arcs whose symbols represent line breaks are removed from the

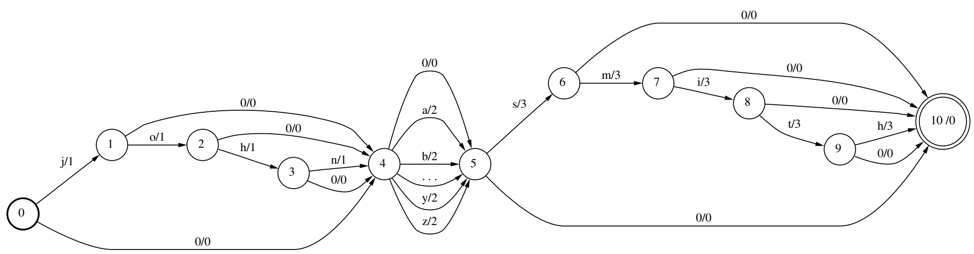
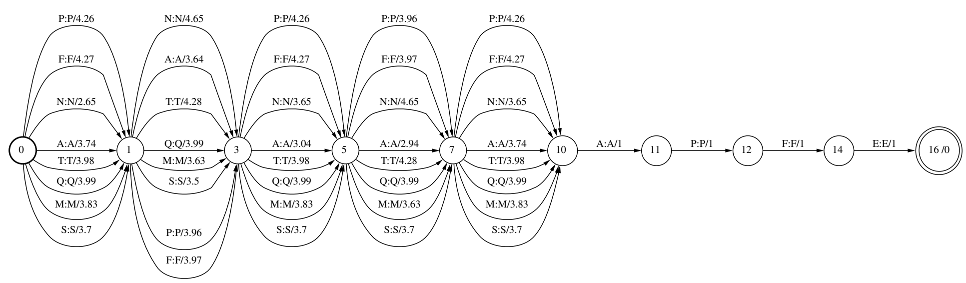


Fig. 15. Well-formed user name FST for “John Smith”.

```
John W. Smith
Rm. 2D-510
Bell Laboratories
700 Mountain Avenue
Murray Hill, NJ 07974
Tel: (908) 582-3433
Fax: (908) 582-7308
e-mail: jws@bell-labs.com
```



E: email address
W: web address
P: phone number
F: fax number
N: personal name

A: postal address
T: title
Q: quote
S: stub
M: miscellaneous

Fig. 16. A reading block and its input WFST.

WFST in the figure for ease of reading.) This WFST is called *input WFST*, since it is the first in a cascade of WFSTs. Note, that, although in this example the input and output symbols on each arc are identical, they could be different due to encoding in the language-directed segmentation algorithm to be discussed in Section 5.3.1. Note also that the number of pairs of neighboring nodes is equivalent to the number of line segments in the reading block, not the number of lines, because a line may be divided into several line segments. In Figure 16, the fifth line in the reading block is divided into two line segments.

It is worth noticing that cost estimation is context free. It tries to estimate the costs of relating the line segment to various functional classes as if it appears alone without any preceding or following line segments in a signature block. We try to make the estimate optimal locally, but it may or may not also be optimal globally, when the context is taken into account. For example, in Figure 16, for the line segment "`John W. Smith`" the personal name functional class has the lowest cost; however, for the line segment "`Murray Hill`" the true functional class (address) is not the one with the lowest cost.

This ambiguity is resolved by contextual knowledge, as incorporated in the subsequent lexicon and grammar WFSTs. For example, the fact that an address is usually composed of multiple line segments, while a personal name is composed of a single line segment is represented, and a penalty is applied for violating it in the lexicon and grammar WFSTs. The functional class of a line segment is finally determined by finding the global optimum under both the context-free cost estimations (represented in the input WFST) and the contextual information (incorporated in the lexicon and grammar WFSTs). For example, in Figure 20, which is the globally optimal path for Figure 16, "`Murray Hill`" is correctly identified as an address.

5.3.1 *Language-Directed Segmentation*. Oversegmentation errors resulting from geometric analysis cause serious problems for cost estimation of line segments. A pattern in an entire line segment may not be retained by its subsegments. For example, while "`John Smith`" is identified as a personal name with regard to the user name "`jws`" by the personal name identification algorithm, neither of the first name or family name alone can be identified in this way.

Since the oversegmentation problem cannot be solved by geometrical analysis alone, a language-directed segmentation approach is proposed. For all the line segments on the same line in a reading block, all possible segmentation positions are evaluated. In other words, we try to combine any two or more adjacent line segments on the same line into a new line segment, and all the possible combinations are built into the input WFST. Therefore, the input WFST contains choices for not only functional class of each line segment but also segmentation positions on each line of the reading block. The best choices of both of them are to be determined together after the input WFST is composed with the lexicon and grammar WFSTs.

For example, consider the text line "`Dr. John W. Smith`". Since the words are written very far apart, this line is broken into four line segments during the geometrical analysis, where each line segment contains only one word. In order to determine the best segmentation positions, the WFST in Figure 17 is built, which enumerates all possible combinations of the four line segments (represented as `A`, `B`, `C`, and `D` respectively in the figure). Note that each arc in Figure 17 represents several actual arcs, where each actual arc is associated with a different functional class and its associated cost.

5.3.2 *Encoding*. After the input WFST is composed with the lexicon and grammar WFSTs, a best-path search is performed to find the functional class of each line segment (or combination of line segments). In order to trace back the segmentation positions, i.e., the combination of line segments, the input symbol in the input WFST must be encoded to contain

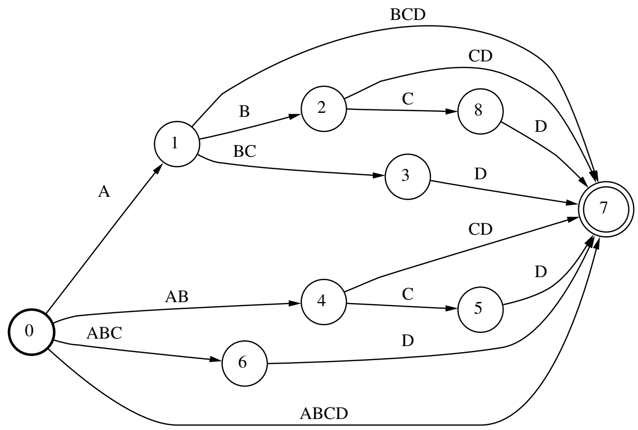


Fig. 17. Input WFST incorporating segmentation choices.

information on both functional class and number of combined line segments. Let K be the number of functional classes; the actual input symbol used is computed as follows:

$$\text{input symbol} = \text{index of functional class}$$

$$+ (\text{number of combined line segments} - 1) \times K$$

The output symbol of the arc need not be encoded, as this is just the index of the functional class.

For example, assume that the indices of functional classes for email address, Web address, telephone number, fax number, personal name, postal address, title, quote, stub, and miscellaneous text are from 0 to 9 respectively. If an arc represents the combination of three line segments and is related to a personal name, its input symbol is $4 + (3 - 1) \times 10 = 24$, and its output symbol is 4.

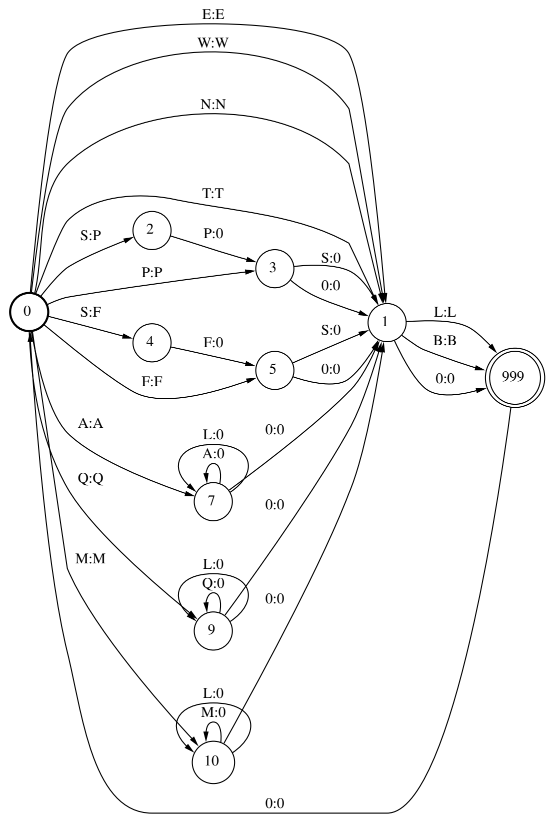
After the best-path search in the composed WFST, each input symbol is decoded to recover the functional class and the number of combined line segments by the following:

$$\text{index of functional class} = \text{input symbol MOD } K$$

$$\text{number of combined line segments} = \text{input symbol DIV } K + 1$$

5.4 The Lexicon WFST

The lexicon WFST describes the construction of a functional block from line segments (Figure 18). For example, a complete postal address could be composed of one or more lines, where each line could, in turn, be composed of one or more line segments. However, a personal name is not usually written in more than one line. Such observations are incorporated in the



E: email address
W: web address
P: phone number
F: fax number
N: personal name

A: postal address
T: title
Q: quote
S: stub
M: miscellaneous

Fig. 18. Lexicon WFST.

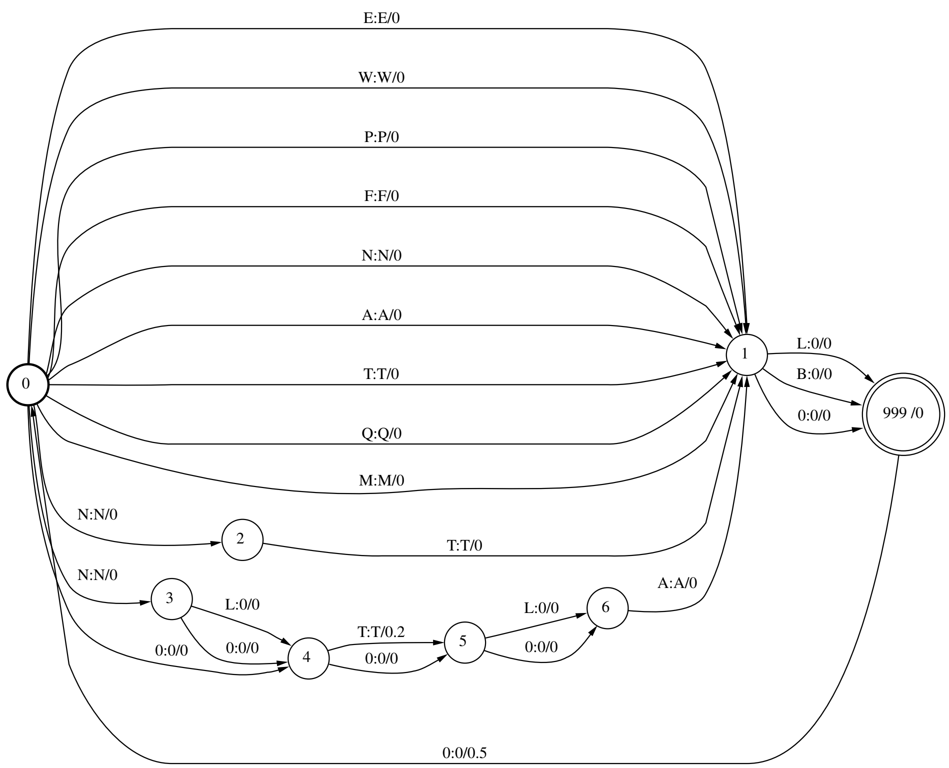
lexicon WFST. As we have noted previously it is at this stage that errors of the kind instantiated in Figure 16 are corrected.

5.5 The Grammar WFST

The grammar WFST describes the construction of a reading block from functional blocks, as shown in Figure 19. To discourage transitions between different lexical units, a moderate cost is assigned to the back-loop transition. We decided to separate the lexicon WFST from the grammar WFST because they represent different levels of abstraction. Furthermore, it will make our future work on the N-gram functional block model easier (Section 8).

5.6 Determination of Functional Classes and Segmentation Points

To determine the functional class of each line segment as well as the segmentation positions on each line, the input WFST is composed with the



E: email address
W: web address
P: phone number
F: fax number
N: personal name

A: postal address
T: title
Q: quote
S: stub
M: miscellaneous

Fig. 19. Grammar WFST.

lexicon and grammar WFSTs. By examining the optimal path in the final WFST, adjacent line segments relating to the same functional class are grouped into one functional block, and therefore a reading block is broken into several functional blocks. Figure 20 shows the global optimal path in the composition of the input WFST from Figure 16, the lexicon WFST (Figure 18), and the grammar WFST (Figure 19). Note that paths with no input symbol, no output symbol, and zero cost are pruned from the figure, for clarity.

6. SIGNATURE BLOCK IDENTIFICATION

A different version of the algorithms for geometrical and linguistic analysis can be used for the *identification* of signature blocks. In the email text-to-speech rendering system, signature blocks are identified from the email

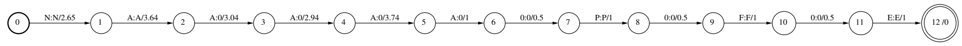


Fig. 20. The global optimal path for the reading block shown in Figure 16.

message by an N-gram character class model [Sproat et al. 1998], which is not highly reliable for signature blocks, and a more accurate algorithm is necessary.

It is observed, in the signature block, that we expect to see text representing email address, Web address, telephone number, fax number, name and postal address (nonmiscellaneous text) more often than miscellaneous text, while in the nonsignature block the opposite is often true. It is also observed usually that there are at least two different nonmiscellaneous functional blocks in a signature block. Therefore, by counting the number of different nonmiscellaneous functional blocks and comparing the ratio of the lengths of miscellaneous text and nonmiscellaneous text in a candidate signature block, identification is achieved.

7. EXPERIMENTS

Experiments were carried out on both signature block analysis and identification. For these experiments, the thresholds used in geometric analysis and the relative costs in the WFSTs were not obtained through formal training due to lack of training data. Rather, they were chosen manually based on common observations of signature samples not used in testing.

7.1 Signature Block Analysis

Signature block analysis was tested on 1361 signature blocks collected from Lucent Technologies, the Department of Computer Science at Concordia University, and various other external sources.[1] They represent a variety of geometrical layouts and writing styles. Neither Lucent nor Concordia has any standard for the composition of signature blocks. There are, all together, 5491 functional blocks in the testing samples, and 97% of them are analyzed correctly. The average speed is 0.58 seconds per signature block on a 150MHz SGI Indy.

The errors can be divided into two categories: geometrical errors and linguistic errors.

7.1.1 *Geometrical Error*. Geometrical errors are unlikely to occur within a reading block, since undersegmentation errors are corrected during cost estimation, and oversegmentation errors are adjusted by the language-directed segmentation approach. However, although rare, there are geometrical errors across reading blocks which are not detectable by our approach. Figure 21 shows a case where one real reading block is mistakenly broken into two juxtaposed reading blocks because they are

[1]These other sources consist of the authors' personal messages which come from sites scattered among major universities and companies in the continental North America and a few foreign countries.

```
(202)530-7024 (Office)
(202)530-7007 (fax)
```

Fig. 21. An example of geometrical error (the bounding box is not part of the text).

```
Frank Soong x2495 2D-436
```

Fig. 22. An example of linguistic error (the bounding box is not part of the text).

very far apart. In the left reading block, there is no keyword to indicate whether the number is a telephone or fax number, which causes an error.

7.1.2 *Linguistic Errors*. Linguistic errors occur when there is no strong lexical pattern in the text. In Figure 22, `2D-436` by itself does not indicate strongly a postal address in and of itself. Since cost estimation is not reliable for these examples and contextual information is lacking, they are not classified correctly.

7.2 Signature Block Identification

Signature block identification was tested on 351 complete email messages from various sources. Emu's preprocessor intentionally overclassifies a considerable number of nonsignature blocks as signature blocks, and our algorithm is then used to pick out the real cases.

The overall recall rate is 53% (97/183), and precision is 90% (97/108). The recall rate may seem quite low. However, among the 86 recall errors, 79 are trivial one-line signatures such as "`-John`" or "`-J`". These trivial signatures are of little interest to most applications, such as the email text-to-speech rendering, and it causes virtually no harm to miss them. The recall rate becomes 93% (97/104) if these trivial signatures are not counted.

8. SUMMARY AND FUTURE WORK

In this article, we describe a new approach that combines two-dimensional structural analysis with one-dimensional grammatical constraints for analyzing the signature block in an email message. The approach consists primarily of geometrical analysis and linguistic analysis, which are intended to deal with geometrical and linguistic complexity of the signature block respectively.

Geometrical analysis converts the two-dimensional signature block into one-dimensional reading blocks. Aside from making the following one-dimensional linguistic analysis possible, it also serves to ensure the coherence of text inside a reading block. Geometrical analysis is done via line segment extraction and connected-component analysis. Some segmentation errors are detected and corrected based on their geometrical properties alone.

Linguistic analysis identifies the functional classes of text in a reading block. This is done by taking into account the lexicon and grammar constraints of the signature block through the use of weighted finite-state transducers (WFST). First, an input WFST representing a reading block is constructed after estimating the cost of relating a line segment with each functional class. Lexicon and grammar constraints are represented, respectively, as a lexicon and grammar WFST, which are composed with the input WFST. The optimal path in the final WFST reveals the identity of each line segment in the reading block.

Most of the segmentation errors which result from the geometrical analysis and cannot be detected or corrected by geometrical information alone are captured and remedied during linguistic analysis. Undersegmentation errors are detected in the cost estimation phase. To deal with the oversegmentation problem, a language-directed segmentation approach is proposed, which incorporates the choice of not only functional classes but also segmentation positions into the input WFST. Therefore, the optimal functional classes and segmentation positions are not detected separately. They are decided at the same time from the optimal path in the final WFST.

A modified version of the algorithms for the geometrical and linguistic analysis is also used for the identification of a signature block. Based on some common observations, a signature block is identified by counting the number of nonmiscellaneous functional blocks and comparing the ratio of the lengths of miscellaneous text and nonmiscellaneous text.

Signature block analysis is currently implemented as a component of an email text-to-speech rendering system. It was tested on 1361 signature blocks containing 5491 functional blocks with various styles from various sources. Ninety-seven percent functional blocks are classified correctly. The recall and precision of signature block identification are 53% (93% if trivial signatures are not counted) and 90% respectively.

As discussed in Section 7, our approach occasionally makes geometrical and linguistic errors. We foresee no general solutions to linguistic errors other than adding more ad hoc rules. However, there seem to be some general solutions to geometrical errors.

Our current algorithm is capable of correcting most of the undersegmentation and oversegmentation errors inside a reading block. However, if the segmentation error occurs across different reading blocks, there is no mechanism to detect or correct it, which is where most of the geometrical errors come from. Figure 21 shows a typical example where a reading block is mistakenly broken into two in the connected-component analysis. Once broken, they will not be reunited.

We are currently investigating several different approaches to solving this problem. One possibility is to try different combinations of all reading blocks and select the combination that yields the lowest overall cost in the final WFST, just as we try different combinations of all line segments on the same line in the language-directed segmentation approach. Alternatively, a different approach to connected-component analysis can be used to generate multiple candidates for reading block segmentation. The challenge is how to efficiently find the N best candidate paths each of which traverses all the line segments once and only once.

We are also considering the use of an N-gram functional block model to enrich the grammar model. However, much more training data are needed for this purpose.

ACKNOWLEDGMENTS

The authors would like to thank David Lewis from AT&T Labs for suggesting the topic, and the reviewers for their constructive comments.

REFERENCES

ANTONACOPOULOS, A. AND RITCHINGS, R. T. 1994. Flexible page segmentation using the background. In *Proceedings of the 11th International Conference on Pattern Recognition* (Jerusalem, Oct.) 339–344.

BAIRD, H. S. 1992. Anatomy of a versatile page reader. *Proc. IEEE 80*, 7, 1059–1065.

BAIRD, H. S., JONES, S. E., AND FORTUNE, S. J. 1990. Image segmentation using shape-directed covers. In *Proceedings of the 10th Internation Conference on Pattern Recognition* (Atlantic City, NJ, June)

DENGEL, A. AND BARTH, G. 1988. High level document analysis guided by geometric aspects. *Int. J. Pattern Recogn. Artif. Intell. 2*, 4, 641–655.

ETEMAD, K., DOERMANN, D., AND CHELLAPPA, R. 1997. Multiscale segmentation of unstructured document pages using soft decision integration. *IEEE Trans. Pattern Anal. Mach. Intell. 19*, 1, 92–96.

GUYON, I., SCHENKEL, M., AND DENKER, J. 1996. Overview and synthesis of on-line cursive handwriting recognition techniques. In *Handbook on Optical Character Recognition and Document Image Analysis* World Scientific Publishing Co., Inc., River Edge, NJ, 1–43.

HOROWITZ, E., SAHNI, S., AND ANDERSON-FREED, S. 1993. *Fundamentals of Data Structures in C*. Computer Science Press, Inc., New York, NY.

JAIN, A. K. AND BHATTACHARJEE, S. K. 1992. Address block location on envelopes using Gabor filters. *Pattern Recogn. 25*, 12, 1459–1477.

MIZUNO, M., TSUJI, Y., TANAKA, T., IWASHITA, M., AND TEMMA, T. 1991. Document recognition system with layout structure generator. *NEC Res. Dev. 32*, 2, 430–437.

MOHRI, M., PEREIRA, F., AND RILEY, M. 1997. A rational design for a weighted finite-state transducer library. In *Proceedings of the 2nd International Workshop on Implementing Automata* (Ontario, Canada, Sept.) 43–53.

NAGGY, S. C. S. AND STODDARD, S. D. 1985. Document analysis with an expert system. In *Proceedings of Pattern Recognition in Practice II* (Amsterdam, June)

NAGY, G., SETH, S., AND VISWANATHAN, M. 1992. A prototype document image analysis system for technical journals. *IEEE Computer 25*, 7 (July 1992), 10–22.

PAVLIDIS, T. 1982. *Algorithms for Graphics and Image Processing*. Computer Science Press, Inc., New York, NY.

PAVLIDIS, T. 1991. Page segmentation by white streams. In *Proceedings of the International Conference on Document Analysis and Recognition* (St. Malo, France) 945–953.

PEREIRA, F. C. N. AND RILEY, M. D. 1996. Speech recognition by composition of weighted finite automata. CMP-LG archive paper 9603001. Los Alamos National Laboratory, Los Alamos, NM. Available via http://xxx.lanl.gov/abs/cmp-lg/9603001

PORTER, G. B. AND RAINERO, E. V. 1992. Document reconstruction. In *Proceedings of the Conference on Electronic Publishing* 127–141.

RAHGOZAR, M. A., FAN, A., AND RAINERO, E. V. 1994. Tabular document recognition. In *Proceedings of SPIE* (San Jose, CA, Feb.) 87–96.

RUS, D. AND SUMMERS, K. 1994. Using white space for automated document structuring. TR-94-1452. Cornell University, Ithaca, NY.

SPROAT, R. 1996. Multilingual text analysis for text-to-speech synthesis. *J. Nat. Lang. Eng. 2*, 4, 369–380.
SPROAT, R., Ed. 1997. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer Academic, Dordrecht, Netherlands.
SPROAT, R., CHEN, H., AND HU, J. 1998. Emu: An e-mail preprocessor for text-to-speech. In *Proceedings of the IEEE Workshop on Multimedia Signal Processing* (Los Angeles, CA, Dec.) IEEE Press, Piscataway, NJ, 239–244.
SRIHARI, S. N. 1987. Recognizing address blocks on mail pieces. *AI Mag. 8*, 4 (Winter 1987), 25–40.
TAKASU, A., SATOH, S., AND KATSURA, E. 1993. A matrix grammar for document processing. In *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* 197–200.
TAKASU, A., SATOH, S., AND KATSURA, E. 1994. A document understanding method for database construction of an electronic library. In *Proceedings of the 12th IEEE Conference on Computer Vision and Pattern Recognition* IEEE Press, Piscataway, NJ, 263–466.
WANG, D. AND SRIHARI, S. N. 1989. Classification of newspaper image blocks using texture analysis. *Comput. Vision Graph. Image Process. 47*, 3 (Sept. 1989), 327–352.

Received: November 1997; revised: May 1998 and February 1999; accepted: March 1999