Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition

Majdi Ben Hadj Ali, Frank Fein, Frank Hönes, Thorsten Jäger, Achim Weigel

German Research Center for Artificial Intelligence (DFKI) - GmbH
Projects OMEGA, INCA, PASCAL2000/PASCAL+
P.O. Box 2080, 67608 Kaiserslautern, FRG
phone: (+49 631) 205 3215
email: {dokana-ma}@dfki.uni-kl.de

Abstract

Document analysis is responsible for an essential progress in office automation. This paper is part of an overview about the combined research efforts in document analysis at the DFKI. Common to all document analysis projects is the global goal of providing a high level electronic representation of documents in terms of iconic, structural, textual, and semantic information. These symbolic document descriptions enable an "intelligent" access to a document database.

Currently there are three ongoing document analysis projects at DFKI: INCA, OMEGA, and PASCAL2000/PASCAL+. Though the projects pursue different goals in different application domains, they all share the same problems which have to be resolved with similar techniques. For that reason the activities in these projects are bundled to avoid redundant work. At DFKI we have divided the problem of document analysis into two main tasks, text recognition and text analysis, which themselves are divided into a set of subtasks.

In a series of three research reports the work of the document analysis and office automation department at DFKI is presented. The first report discusses the problem of text recognition, the second that of text analysis. In a third report we describe our concept for a specialized document analysis knowledge representation language.

The report in hand describes the activities dealing with the text recognition task. Text recognition covers the phase starting with capturing a document image up to identifying the written words. This comprises the following subtasks: preprocessing the pictorial information, segmenting into blocks, lines, words, and characters, classifying characters, and identifying the input words.

For each subtask several competing solution algorithms, called specialists or knowledge sources, may exist. To efficiently control and organize these specialists an intelligent situation-based planning component is necessary, which is also described in this report. It should be mentioned that the planning component is also responsible to control the overall document analysis system instead of the text recognition phase only.

Keywords

Document Analysis, Layout Extraction, Text Recognition, Segmentation, Character Recognition, Voting, Lexical Postprocessing, Analysis Control

Contents

1 Introduction

Work cycles in offices typically include many processes where clerks communicate through different media. As mean for communication printed documents still play the central role in modern offices. Due to the simplicity in generating and interchanging documents, the amount of messages to be processed increases dramatically [Schuhmann 87]. To handle the resulting “information overload”, it is necessary to separate the important key messages from the insignificant data and to present it to the user in a task specific or workflow oriented manner. In this way, each clerk would have the possibility of accessing information by different features (structural, textual, referential), and therefore gets comfortable access to the relevant information. Furthermore, this enables automatic processing of information by other systems.

The Document Analysis and Office Automation Department (*DAD*) at *DFKI* undertakes an attempt towards an integrated handling of information coming into an office by mail, fax and e-mail. The overall goal is to analyze a company’s incoming written communication, to extract and to store relevant information automatically to allow comfortable retrieval or automatic processing by post-ordered systems.

Research at *DAD* started with the *ALV*[1] project in 1990 [ALV 94]. In this project, analysis tools for printed, single-page business letters have been developed and implemented. A major result was a system that is able to transform a letter from its printed form to a symbolic representation which is based on the international *ODA* standard [Dengel 92]. Today there are three projects at *DAD*: *OMEGA*, *INCA*, *PASCAL2000*/*PASCAL+*. The aim of the *OMEGA*[2] project is the integrated analysis of office mail dealing with different relationships to previously analyzed documents. This way, we continue our research started in the *ALV* project consequentially. The other projects are grouped around the *OMEGA* project within the *DAD*. Each of it focuses on particular application aspects. The *INCA*[3] project intends to provide information about technical reports for information retrieval i.e. for its re-finding in a large database. *PASCAL2000/PASCAL+* is concerned with the problem how to provide blind humans access to printed documents. Typically, non-sighted or visually impaired people have no or only few possibilities to process printed information. The *PASCAL2000/PASCAL+* project tries to overcome this weakness by extracting and processing relevant information which is subsequently presented appropriately. Another major research topic within the *PASCAL2000/PASCAL+* project is technology assessment for document analysis systems within the office environment. This means that also philosophical, sociological, and ergonomic aspects are considered.

Though the projects pursue different goals in different application domains, they all share the same problems which have to be resolved with similar techniques. For that reason the activities in these projects are bundled to avoid redundant work.[4] At *DAD* we have divided our research in document analysis into four main topics:

- Text Recognition
- Text Analysis
- Reactive Plan Generation
- Knowledge Representation for Document Analysis Tasks

Text recognition covers the problem of transforming captured document images into a sequence of

1. **A**utomatisches **L**esen und **V**erstehen
2. **O**ffice **M**ail **E**xpert for **G**oal-directed **A**nalysis
3. automatic **IN**dexing and **C**l**A**ssification of documents
4. It should be mentioned that we are mainly interested in solutions which can be used in different applications only by making minor modifications.

words in *ASCII*, whereby a correct reading flow should be preserved. The aim of text analysis is to process the word hypotheses and to extract the relevant semantic information. These two research areas are further divided into several subtasks. For each subtask several competing solution algorithms, called specialists or knowledge sources may exist. To efficiently control and organize these specialists in a complex overall document analysis system, a reactive plan generation component is necessary. Generally, analyzing a document is a very complex and challenging task, which uses a huge amount of knowledge. To efficiently represent and process this knowledge a specialized knowledge representation formalism combined with optimized inference mechanisms is of great benefit.

The activities within the *DAD* at *DFKI* will be published in three different research reports. This one focuses on text recognition which has the aim of transforming binarized document images into machine readable (words encoded with *ASCII*) form. Furthermore, we also describe our activities in reactive plan generation, but it should be mentioned that plan generation is also necessary for text analysis. The second report [Baumann 95] discusses all aspects of text analysis for the extraction of document semantics and in [Bläsius 95] the underlying knowledge representation is presented.

The report at hand is organized as follows. In Section 2 we describe the subtasks of text recognition along with our proposed solutions. The subtasks are image preprocessing, layout extraction, font attribute identification, OCR, voting of OCR results, and lexical postprocessing. Furthermore, in Section 2.6 we describe the task of reactive plan generation. The overall system architecture as well as the current state of realization of the text recognition module is described in Section 2.6. Section 4 is dedicated to the fundamental problems of testing and performance measurement. We finish the report with some concluding remarks in Section 5.

2 Text recognition tasks

The task of text recognition is to transform the binarized image data containing textual information into its “usual” representation as strings over a given alphabet (e.g. ASCII). Research in text recognition, especially in OCR, takes place since the early’ 50s. In the last decades, a tremendous progress in recognition accuracy has been achieved. Today’s commercial OCR devices reach high reliable results on good quality documents [UNLV 94]. However, on low quality printings as they appear in daily office work, current devices are far from recognition accuracies demanded for subsequent automatic text analysis. The most frequent OCR errors are simple character substitutions (e.g. ‘e’ -> ‘c’), improper segmentation of characters resulting in multi-character substitutions (e.g. ‘m’ -> ‘rn’), and unrecognized characters [Esakov 94]. Additionally, errors may also occur on the word level, e.g. by the use of incomplete dictionaries or improper word segmentation. Thus, text recognition with respect to the demands of automatic text analysis can still be seen as a challenging task.

In our *DAD* we plan to develop a text recognition module which provides as good as possible recognition results on the word level with respect to the demands of the text analysis tasks. Following the “traditional” approach, all characters on a document page should be identified and classified as a member of a given alphabet. Furthermore, characters should be merged to words, text lines and subsequently to a reading flow. Because for text analysis the word level is the most interesting one[5], a verification of character sequences against legal words plays a central role.

As mentioned in the section before our research activities in document analysis are divided into a text recognition part, a text analysis part, a reactive plan generation part, and a knowledge representation part. In this section the main text recognition subtasks are introduced and strategies for their solutions are presented. Following subtasks are considered:

- Image preprocessing and layout extraction
- Font attribute identification
- Character classification (OCR)
- Voting of OCR results
- Lexical postprocessing

For each recognition subtask, there may be several competitive or cooperative problem solving algorithms. The set of all problem solving algorithms together with a reactive plan generation module, including different recognition strategies, form our text recognition approach. Figure 1 illustrates the recognition modules and their relations. There is no fixed activation sequence of analysis algorithms. The corresponding sequence is determined by the reactive plan generation module depending on the current situation.

5. beside the reading flow

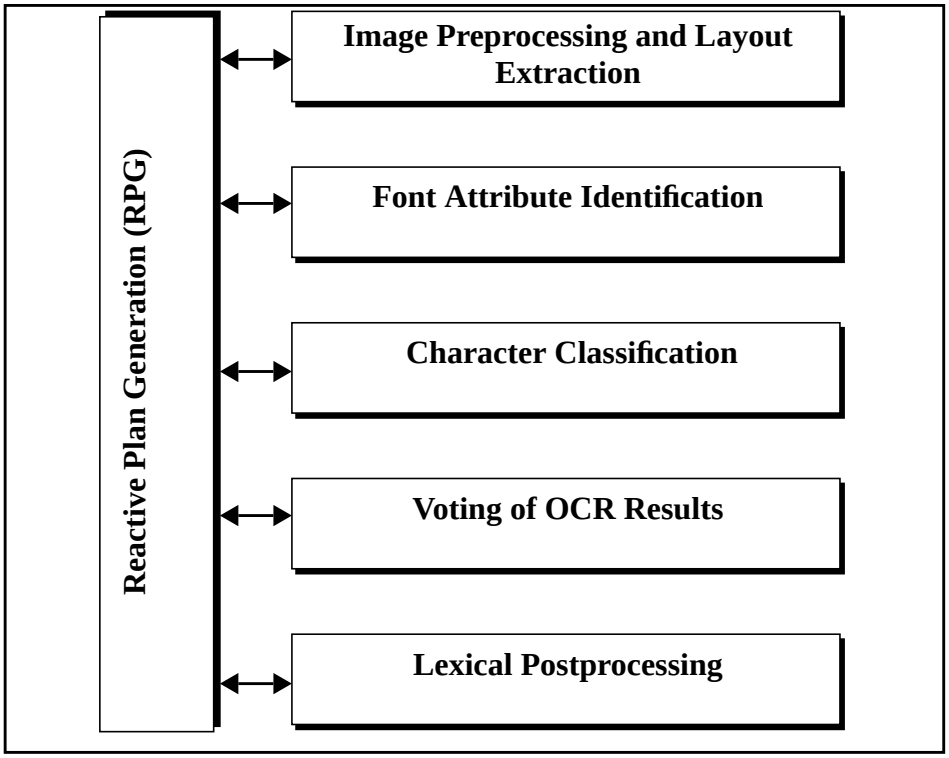

Figure 1: Text recognition components.

2.1 Image preprocessing and layout extraction

The purpose of document image analysis is to transform the information contained on a digitized document image into an equivalent symbolic representation. Text parts are the main information carrier in most applications. For that purpose, it is necessary to locate text objects within the image, recognize them, and extract the hidden information.

The increasing number of comfortable publishing systems available today means that documents may contain, beside text, also graphics and images which overlap each other (see Figure 2). Additionally, text lines may not be horizontally aligned. Therefore, finding text parts and locating characters, words, and lines are not trivial tasks.

Figure 2: Image of an advertisement.

In this section, we present a robust algorithm for separating textual information from non-text and for grouping text elements into characters, words, lines, and blocks. The algorithm is completely independent from text orientation and is able to process text in various fonts and styles. Both black characters on white background as well as inverse printing can be nested in graphics.

A number of techniques for image processing and layout extraction have been proposed in literature [Nadler 84]. Before describing our own algorithm, we want to discuss existing procedures in order to differentiate our proposal from others. The techniques can be grouped into three different categories: systems only separating text from non-text, systems segmenting characters, words, and lines, and approaches which consider both tasks.

Representatives of the first category are [Wahl 82] and [Wang 89] which classify areas of a document as text, graphic, or image. The basic objects are blocks which do not overlap each other and contain only one mode of information. Text objects are only horizontally aligned. In [Bartneck 89] some elementary image processing techniques are described, which classify single image points (pixels). The aim of these approaches is to reduce image noise and thus separate noise from text.

In the second category, there are a large variety of approaches. A top down algorithm is presented in [Nagy 85] segmenting a document by a set of vertical and horizontal line segments. A global to local approach for segmenting Manhattan[6] layouts into columns is described in [Baird 90] considering the structure of the background (white spaces). In [Higashino 86] a rule-based approach for top-down analysis of document layouts is presented where a specific model exists for each document class. All approaches in this category consider only horizontally aligned text. If an area cannot be divided into horizontal lines, it is classified as non-text.

Approaches combining text/non-text classification with segmentation make up the third category. For example, a rule-based approach is introduced in [Fisher 90]. Connected components of a smeared image are classified as text or non-text and afterwards, they are grouped into words, lines, and blocks. Due to the smearing process, split characters can be often reconstructed, but at the same time, neighboring characters as well as characters and non-text components are often merged. Because the parameters for smearing are global for the entire document, problems may occur if there are characters in different font sizes. Fletcher and Kasturi [Fletcher 88] propose another technique where eight connected black components are the primitives. After a filtering phase, a Hough

6. The subclass of Manhattan layouts can be segmented by a set of vertical and horizontal line segments.

transformation is performed which groups objects associated with a particular line. All objects belonging to a line are grouped into strings and designated as text. This approach is able to analyze different font sizes in one image, but there is no phase handling split characters. Another symbolic approach using the same primitives is described in [Bixler 88]. In the first step, a filtering is performed where knowledge about the exact font size of the document is necessary. Afterwards, strings are generated while analyzing the neighborhood of each component. Thus, the approach is only able to process text in pre-specified sizes. As in the two former systems, this algorithm is also based on the assumption that each character corresponds exactly to one connected component and therefore, character splitting is disregarded. In [O'Gorman92] a self adaptive algorithm is proposed that uses the document spectrum *docstrum* consisting of distances and angle measurements between connected components. Components that cannot be grouped into words or lines are designated as non-text. Text strings can have any orientation and different font sizes, but there is only a one to one relationship between characters and connected components. All approaches mentioned above interpret black pixels as image information, white pixels as background. Therefore, they are not able to analyze inverse text.

These systems process different input and produce different results. They operate quite well for specific document classes. Most of them only operate on horizontally aligned text. None of them deal with characters which are split into several connected components and none consider normal as well as inverse text with different orientations in one analysis phase.

After this brief overview of existing approaches, the next subsection describes the progress of our approach with respect to earlier versions as described in [Hönes 92] and [Hönes 93]. It illustrates the single analysis phases of our system in detail and specifies the classes of documents we can process. The following subsection illustrates experimental results and discusses strengths and weaknesses. Section 2.1.3 concludes with a summary of the main characteristics of our approach and with our planned activities.

2.1.1 An algorithm for text extraction and segmentation

Before starting to describe our algorithm we should state which kind of documents we want to analyze. Instead of selecting a special class, we itemize several requirements often found in documents, such as modern business letters. A layout extraction procedure as part of a document analysis system should be able to process documents with the following features:

- Text may be written in different font styles and sizes
- Characters may be split into two or more connected components (e.g Ä, ö, i)
- Text strings may have any orientation
- Different text strings may have different orientations
- Text and non-text regions may be nested
- Text strings are approximately line-oriented
- Normal as well as inverse printings may occur

Our robust and efficient algorithm for text extraction and segmentation is able to process documents with the above described features. It separates text elements from non-text ones, regardless of the line orientation and font size or style. Text elements are grouped into characters, words, lines, and blocks. In addition the skew of each text line is calculated in order to deskew it for providing best results in a post-ordered text recognition phase. Instead of applying commonly used time-consuming algorithms, we developed simple heuristics specially adapted to specific characteristics of text.

The algorithm performs no character recognition. Text extraction and segmentation is only based

on the size and arrangement of the connected components. For proper operation, only few restrictions exist:

- Text and non-text pixel groups should not merge.
- The maximum text font size should be less than 24 points (system parameter).
- The distance of parallel text strings should not be smaller than the average intercharacter gap.
- Generally, text of a line should be aligned along a common orientation (approximate straight line).
- Every text line should consist of at least three characters.
- Text parts should contain noise under a certain threshold.

The system performance will be highly reliable, if these constraints are satisfied. However, if some of them are not completely satisfied, the algorithm still produces acceptable results. For example if the inter-line spacing is smaller than the inter-character spacing, the algorithm operates correctly unless the block consists of three or more lines.

The algorithm we introduce has some similarities to a few well known techniques, but differs in three important aspects from other approaches. It is able to handle inverse text, it adapts itself on the current font size, and it is able to handle broken or split characters up to a certain degree.

Our analysis is divided into 9 consecutive phases:

- Connected Component Analysis
- Filtering
- Neighborhood Determination
- Temporary Line Generation
- Relaxation
- Text/Non-text Classification
- Grouping to Characters
- Grouping to Words
- Grouping of Non-text

In the following sections we explain each single phases in more detail.

2.1.1.1 Connected component analysis

For connected component analysis we use an algorithm called SPRLC[7], developed by [Mandler 90]. It is a very efficient algorithm that generates a hierarchy of four-connected black respectively eight-connected white components. All connected components — independent from their color — can be classified as text. Therefore, the image is considered as a hierarchy of several layers where the lowest level represents the white document page and the highest, e.g., a black i-dot. As final representation, we obtain a tree representation where each node is a connected component and the root represents the entire page. Figure 3 illustrates the layers on a sample image for the letter “B”.

7. a German acronym for single-pass contour line coding

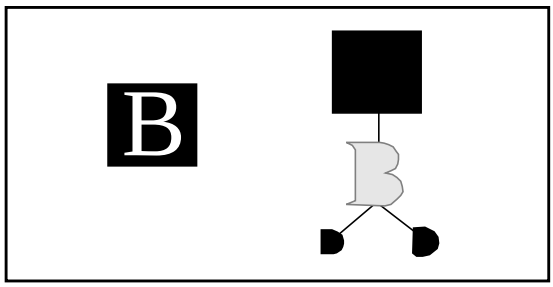

Figure 3: Letter “B” as a hierarchy of connected components.

Because black and white connected components are treated equally, i.e. are analyzed in the same way, the basis for handling inverse text is provided.

2.1.1.2 Filtering

Typically, documents we consider have a resolution of 200 to 300 dpi[8] and consist of 1.000 up to 10.000 connected components. Many components are very small and few are very large. If we were to analyze all these objects together and their relationships to each other, it would be far too time consuming.

The aim of this phase is to reduce the huge set of connected components to those which are likely to be text. Instead of performing a reliable filtering, we only want to extract the main components of a text line and mark the others as non-relevant for the next two phases. Our filtering uses only a very simple and easily-computable feature: the average of object width and height. The filter marks each connected component as non-relevant if its average of width and height is larger than $size_{max}$ or smaller than $size_{\min}$, where $size_{\min}$ and $size_{\max}$ are system parameters. In our tests we filter objects larger than 60 points and smaller than 6 point characters. Thus the number of components is reduced to about one third.

The filtering is very rough, i.e. some text object may be lost and some non-text ones may pass the filter. However, potential defects will be restored in later analysis stages (see Inverse filtering).

2.1.1.3 Neighborhood determination

The aim of this phase is to determine all neighbors of each connected component. Neighbor means that the current object and a neighbored object possibly belong to one and the same line — more precisely one follows the other in the same line.

A simple method for neighbor determination is to compare each object with all other components in the document and mark as neighbors those having a small distance to each other. But analyzing documents with more than 1.000 objects is too time-consuming. Therefore we introduce an efficient heuristic.

In a first step we generate a two-dimensional object space. The size of that space depends on the size of the input document. For example for a German standard letter a space of 400x280 elements is generated — independent from its resolution. Each space element can correspond to one or more connected components where the relative position of the center of a component in the document image determines the corresponding space element. Additionally, each element has references to the objects it represents. Figure 4 shows a simplified example of an image and the corresponding object space.

In the second step using this data structure we determine all neighbors of each object. For that

8. dots per inch.

purpose, we define a local neighborhood for each connected component which depends on the size of the connected component (arithmetic average of width and height). Consequently, the neighborhood relation is not symmetrical.

Now, the neighborhood area for every object is combed and the contents of all non-empty cells in the neighborhood are added to the neighbor list of the corresponding connected component. Because only objects having the same color can be neighbors, black as well as white text strings can be analyzed at the same time. Figure 4 also shows the neighborhood for a connected component and the corresponding neighbors.

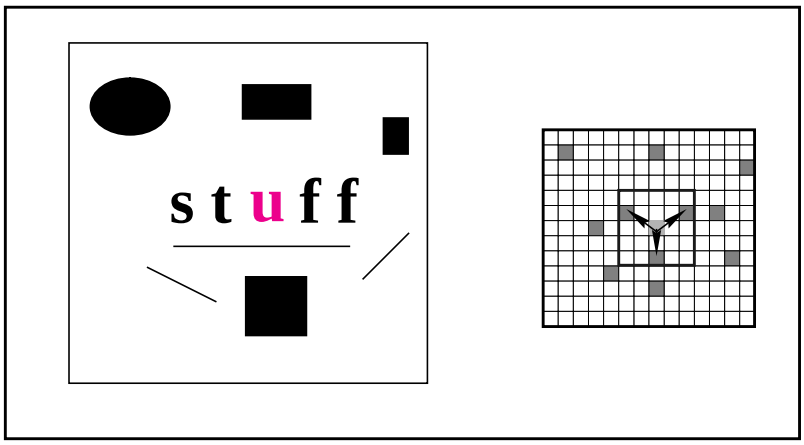

Figure 4: Simple image with connected components and corresponding objects space with neighborhood for the letter “u”.

The procedure proposed is a simple but very efficient heuristic for determining a local neighborhood for text objects. Instead of considering all objects and comparing them with each other, we can reduce the comparisons to only a few relevant objects close to the current object. The exact distance between objects is calculated considering the coordinates of the centers of the components in the original image. Based on this distance, the neighborhood lists are sorted in an increasing manner.

The output of this phase provides a list of neighbors for each component including additional information about the arrangement of the neighbors (angle between components). This information is a basis for the next two analysis phases and guarantees their reliable performance.

2.1.1.4 Temporary line generation

Before we start to describe the temporary line generation procedure we have to define the term *line*. In our system we consider a text line as a sequence of at least three connected components with a common orientation. The maximum distance between these components must be less than D_{line}**average component size* of the text line where D_{line} is a system parameter which has typically the value 3. Every connected component can only belong to one line, but in this phase it is permitted that it can be attached to two or more temporary text lines.

The task of establishing temporary lines is divided into the three subtasks *finding a line anchor*, *calculating the expansion area*, and *line expansion*. A line anchor is used as a starting point for generating a temporary line which is gradually expanded using the expansion area. Only components completely lying within this area can expand a line.

As line anchor, we denote a triplet of components which fulfill the following requirements:

- they are neighbors,

- their distance is less than D_{line}**their average component size*,
- the difference between the two angles is less than $Angle_{\text{AnchorMax}}$,
- they have approximately the same size, and
- at least one of them does not belong to a line.

For finding line anchors, the complete set of connected components is combed. If such a triplet is found, a temporary line is generated which will be subsequently completed. Afterwards, the rest of the set is combed until no more anchors are found.

As mentioned above for line expansion, we have to define an area in which all components have to lie in order to belonging to that line. For that purpose, the orientation of each line is determined using the regression line [Hainzl 85]. The group of components of a line is interpreted as a set of measure points of an experiment. The measured values are approximated by the straight line

$$y = ax + b$$

which includes the smallest square error. The single coefficients are

$$b = \frac{1}{d} \cdot \left(\left(\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i \right) - \left(\sum_{i=1}^{n} x_i y_i \cdot \sum_{i=1}^{n} x_i \right) \right)$$

$$a = \frac{1}{d} \cdot \left(n \cdot \left(\sum_{i=1}^{n} x_i y_i \right) - \left(\sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i \right) \right)$$

where $$d = n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2$$

For calculating the expansion area, we additionally use the line height with respect to the orientation calculated above. The original text line area is expanded on all directions, i.e. on top and bottom with half of the line height and on left and right with D_{line}**average component size* of all line objects. Figure 5 shows a line expansion area for an example text line.

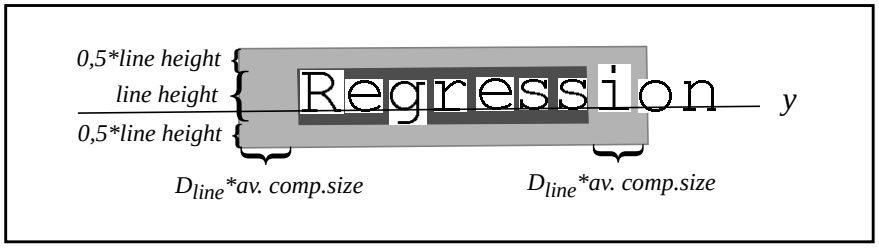

Figure 5: Example of an incomplete text line with expansion area.

The aim of *line expansion* is to complete the initial line, expanding it on both ends using a *continuation condition*. This condition restricts the addition of an object to a line: only components being neighbors of an end object[9] of the line, having similar size, and which lie completely within the expansion area are added. After line expansion, all line features are re-calculated including the expansion area. Line expansion ends if no more adequate components are found.

9. The elements of a line are sorted by the line orientation. End object means in this case the left-most or right-most one.

2.1.1.5 Relaxation

It is very difficult to generate correct and complete text lines without a priori assumptions about the text structure and orientation. The former processing phase yields valuable information about line orientations, but sometimes it includes ambiguities, gaps or errors. Figure 6 illustrates a possible result of the temporary line generation and the expected result. To correct and complete this result we have inserted the relaxation phase.

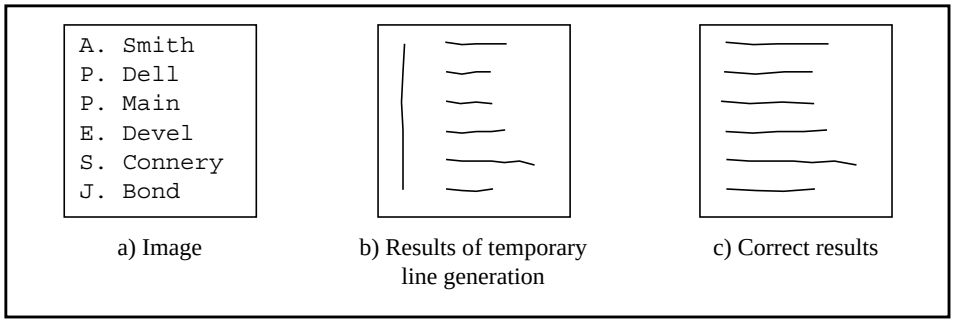

Figure 6: Problems caused by unfortunate line anchor determination.

The relaxation phase consists of four processing steps which are performed successively: *line generation*, *block generation*, *inverse filtering*, and *consistency check*. All four analysis steps make up one relaxation step which can be repeated several times. As a result complete and correct text lines are generated which are additionally grouped into text blocks. Figure 7 illustrates the single analysis steps of the relaxation process.

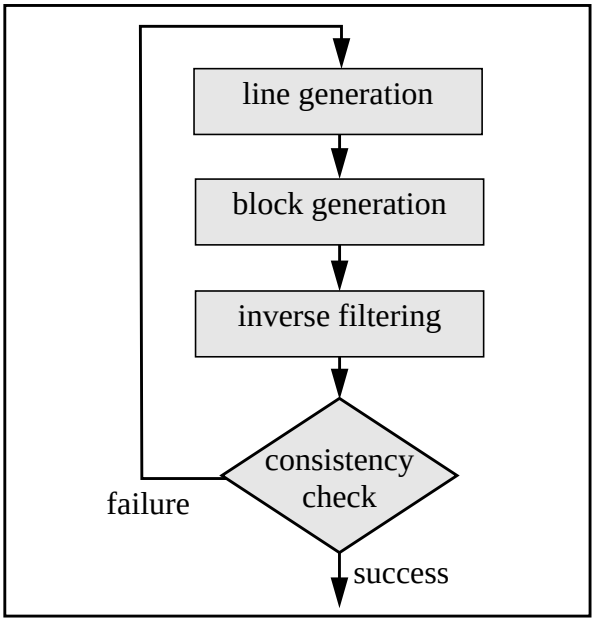

Figure 7: The four analysis steps of the relaxation phase.

The single analysis steps are described in the following sub-sections.

2.1.1.5.1 Line Generation

At this analysis stage we deal with temporary and regular text lines. Caused by larger gaps a line can be split into several sublines. In this step we want to merge two text lines in order to build the longest possible lines. The following conditions must be satisfied for a merge operation:

- Both orientations must be approximately similar (parallelism).
- Both lines have to belong to the same or to no block segment (block membership).
- Their distance must be smaller than a given threshold value (proximity).
- Their heights must be approximately similar (font size).
- Their vertical overlapping relative to their orientation must be large enough.

If two lines fulfill these conditions, a new one is generated including all components of the two original lines. Subsequently, all features of the new line have to be calculated. The line generation step is recursively applied and ends if there are no more pairs of lines satisfying the above conditions.

2.1.1.5.2 Block generation

For verifying text lines a more global context is necessary. For that purpose we group text lines with homogenous appearance to blocks. The larger the text block, the more significant the context. For grouping text lines into blocks following procedure is applied:

1. Choose a text line not belonging to any block.
2. Examine existing blocks, if
   - the current line and the block lines have the same orientation
   - the distance between block and current line is small enough
   - the horizontal overlapping of the block and the line relative to their orientation is large enough
3. If there is a text block satisfying these conditions, group the current line to this block.
4. If there is no such block satisfying the conditions of 2., create a new one.
5. If there are further text lines not belonging to a block, go to 1..

The same procedure is applied to two blocks in order to merge them.

2.1.1.5.3 Inverse filtering

Unfortunately, the lines generated in the above processing phases are not complete. This fact is due to three possible errors:

- It is possible that in phase 2 (Filtering) small text objects representing punctuation marks or parts of split characters are erroneously filtered.
- Small non-filtered text components (e.g. big i-dots or commas) are not assigned to lines, because they have lain outside of the expansion area.
- Larger components representing merged characters are filtered.

This phase aims to remove the first two problems picking up the erroneously filtered components as well as small non-filtered objects lying within the current line scope. For that purpose we consider every small component and check, whether there is another connected component with the same color in its neighborhood belonging to a text line. If the component lies within the line expansion area, the current component is attached to the line. For the neighborhood determination we use the two-dimensional object space described in section *Neighborhood Determination*. The third problem is also solved by implanting larger objects in existing text lines, unless they lie completely within the line scope.

It is important that we distinguish two different kinds of line memberships. The first one, mainly established during temporary line generation, attaches an object as a main component to a line. We

say: a main component expands the line. The other kind of connected components that are mainly considered in this analysis phase are secondary line elements which complete a text line. These elements have a relatively small size. The distinction in these two kinds of line elements is used in Section 2.1.1.6

2.1.1.5.4 Consistency check

The former analysis steps have the goal to generate or to complete text lines and blocks. For these tasks local consistency constraints are used, however global consistence cannot be guaranteed. For example a wrong line anchor can be correctly expanded, but always leads to an incorrect text line. For that purpose in this analysis stage we perform a consistency check which detects possible errors and removes them restoring a valid former situation. Strictly speaking a correction is not performed. Instead inconsistencies are dissolved.

Up to now only text lines and blocks are generated. Therefore the verification is restricted to these two types of objects. In the following we list the features and rules applied.

Text lines:

- If a connected component is higher than $MaxCCHeight_{Line}$ * *average-element-height(line)*, then remove that component from the text line.
- If the ratio of the number of main line elements to secondary ones is larger than $Ratio_{MCSC}$, dissolve the entire text line.
- If a connected component belongs to more than one text line, remove all binding except to those with the best assessment value.

Text blocks:

- If a line is higher than $MaxLineHeight_{Block}$ * *average-line-height(block)*, then remove that line from the text block.
- If a line differs more as $DiffLine_{Block}$ degree from the *average-line-orientation(block)*, remove it from the block.
- If the distance between two neighboring lines is larger than $MaxLineDist_{Block}$, * *average-line-distance(block)*, then divide the block between these lines into two sub-blocks.
- If the percentage overlap of two neighboring lines is less than *MaxOverlLine*, then divide the block between these lines into two sub-blocks.

The verification rules for lines use a line assessment value which reflects the quality of a text line. Because it depends on the assessment value of the corresponding block, we first have to specify block assessment and then line assessment.

The quality of a block depends on the following features:

- Number of text lines
- Number of main components of the longest line
- Biggest difference between main component height and average component height of the block
- Biggest difference between a line orientation and the block orientation

The quality of each text line depends on:

- Ratio of main components and secondary components
- Number of main components

- Assessment value of the corresponding block
- Largest difference between main component size and average main component size of the line

The single application of the first three analysis steps of this phase would not produce satisfactory results. Only the repeated application of all relaxation steps generates correct and consistent block and line segments which again serves as a basis for the following word and character segmentation. Figure 8 and Figure 9 show for the document in Figure 2 the block and line segments after relaxation phase.

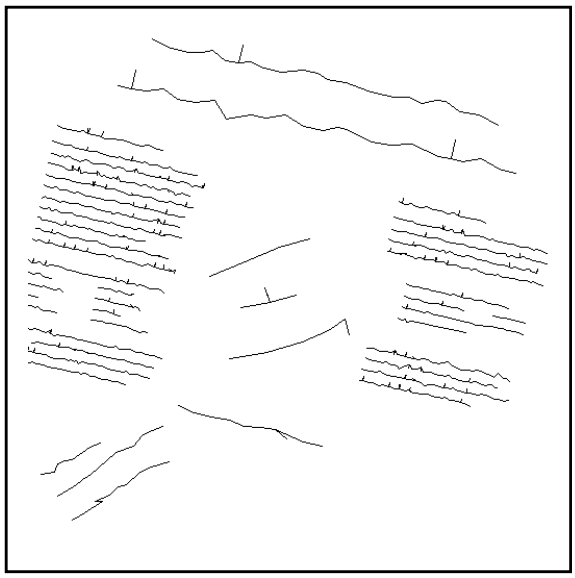

Figure 8: Text lines of Figure 2 after relaxation phase.

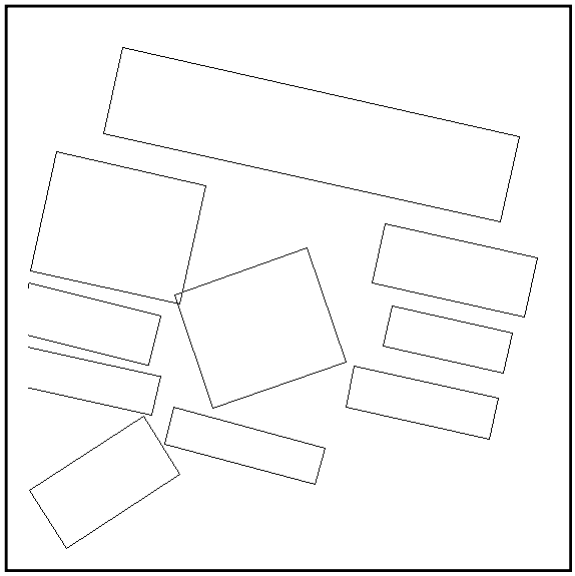

Figure 9: Block segments of Figure 2.

2.1.1.6 Text/non-text classification

Typically, common documents do not only consist of textual constituents. Graphics and photo elements are often combined with text. For a subsequent text recognition these non-text elements are bothersome. They cannot be correctly classified and require processing time. Therefore, the goal of

this analysis phase is to classify connected components as text or non-text ones, because for most applications it is sufficient to distinguish between these two information types.

As a basis for classification as text and non-text connected components are used. Depending on local and global features of a specific component, a text or non-text label is attached. The following local features are used:

- Object size ($\frac{width + height}{2}$)
- Complexity of the object (number of components contained as well as degree of nesting)
- Kind of line membership (main element or secondary element)

As global features the assessment values of blocks and lines (see Section 2.1.1.5.4) are used.

From the variety of classification techniques referred to in the literature (see [Schürmann 77]) we have preferred the simple but highly adaptable method of arithmetic mean of weighted votes:

$$Quality_{text}(CC) = \frac{\sum_{i=1}^{n} (w_i \cdot q_i(CC))}{\sum_{i=1}^{n} w_i}$$

n represents the number of features, w_i the weight of the *i*th feature, and $q_i(CC)$ the probability to be text according to the *i*th feature ($0 \le q_i(CC) \le 1$, where $q_i=0$ denotes non-text and $q_i=1$ text). $Quality_{text}(CC)$ yields an overall rating for the classification as text. Depending on the specific demands and on the current domain, a threshold value is specified calculating the final label.

2.1.1.7 Grouping to characters

In this analysis phase connected components are grouped into characters. In many cases characters consist of only one component. Thus the corresponding characters can be easily established. But there are also several characters which are composed of two or more components. Typical representatives are *Ä, Ö, Ü, ä, ö, ü, i, ?, !, :, ;* and *%*. Additionally in low quality printings, further characters are split into two or more components.

The tasks of character grouping can be transformed into the task of determining which components should be grouped into one character and which belong to others. As grouping criterium we consider the overlapping rate of neighboring components — more precisely the percentage overlapping rate of the circumscribing rectangle parallel to the line orientation. An overlapping rate of 100% of component A and component B states that A completely lies within the scope[10] of B. Because the overlapping relation is not symmetrical, we calculate the largest one, i.e. $max(overlap(CC_i,CC_j),overlap(CC_j,CC_i))$, where i≠j. If two components overlap each other more than *CharOverl* %, both build a joint character, otherwise they belong to different characters.

In the inverse filtering step for each secondary line element the nearest main component within the line was determined. Therefore, the distances from the centers of the circumscribing rectangles parallel to the line orientation were calculated. In this step the percentage overlapping of the two elements are calculated. If it is smaller than *CharOverl* % the two neighbors of the main element are investigated. If none of them overlaps more than the given threshold the secondary element builds a separate character[11], otherwise it builds together with the corresponding main component one single character. The results of these phase are illustrated in Figure 10.

---

10. As scope we understand the overlap along the line orientation.
11. Depending on the component size it could also be interpreted as dirt or noise.

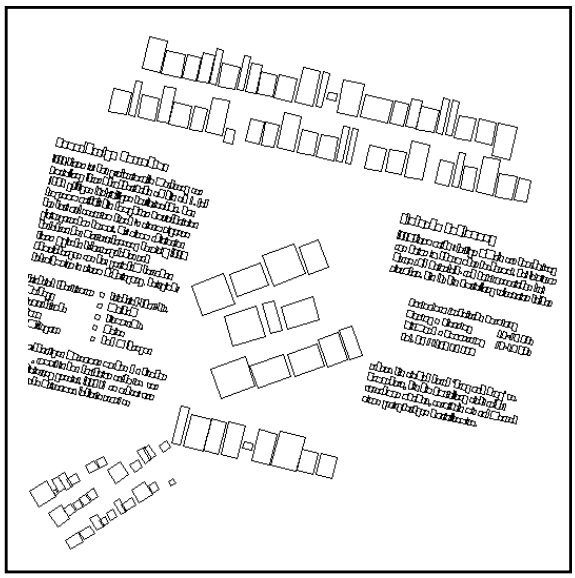

Figure 10: Character segments of Figure 2.

If a character is broken into several parts which do not overlap to a specific degree, our algorithm erroneously generates for each part a character. To solve this problem the integration of a character recognizer would be necessary.

2.1.1.8 Grouping to words

The aim of this analysis phase is to group characters to words. The main criterium for this task is the distance between characters. The original tasks can be reduced to the problem classifying the inter-character distances as character or word separators. Depending on the algorithm for letter and word spacing the two distances or their ratio, respectively vary strongly. A corresponding procedure has to consider this circumstance.

For grouping characters to words, we perform the following analysis steps:

6. Compute a histogram of all inter-character spacings including the frequencies.
7. Evaluate the histogram using the following rules:
   - Identify all zero valleys in the histogram.
   - Consider all valleys occurring beyond *FirstValley* % of the spacings. Ignore all smaller spacings.
   - If there is no such valley, stop evaluating and go to 8..
   - Mark the largest valley and divide the text line into virtual sublines using the distance value of the marked valley.
   - Apply step 7. recursively for the resulting sublines.
     This recursion is necessary, because in each of the lines, words may have different distances and the interpretation of a histogram with several word distances causes problems.
8. Classify unmarked spacing as character separators, marked spacings as word separators.
9. Group characters to words.

This results in a line-oriented word segmentation. It should be mentioned that word hyphenations are not resolved.

2.1.1.9 Grouping of non-text

As a last step of our layout extraction procedure we have to separate text from non-text. For that purpose we inspect the set of connected components and group all non-text elements in a separate block. The corresponding non-text lines are dissolved. Thus, subsequent processing steps may be able to process this block, for example, in order to identify graphical primitives like straight lines or circles.

As final result we have classified connected component as text or non-text. Text components are grouped into characters, words, lines, and blocks whereas non-text components are only accumulated in a separate block.

2.1.2 Experimental results

Having given an introduction to our system design, in this section we evaluate the performance and robustness. We tested about 150 documents[12] with one and the same parameter set. They were digitized with a scanner with a resolution of 300 dots per inch. Afterwards in a first phase, connected components were generated and stored in a file. In a separate phase the layout extraction task was done.

In this section we discuss results of the example images shown in Figure 2, Figure 11, and Figure 12 which represent typical problems and tasks. The documents include text with different orientations, normal and inverse printed characters as well as handprinted character strings. The font sizes vary from 9 to 46 points. There are characters which consist of two or more connected components. Text is also nested in non-text.

As shown in Figure 2, Figure 8, Figure 9, and Figure 10, the algorithm proposed operates quite well. All document parts satisfying the constraints described above are correctly classified and segmented. Even strings of handprinted isolated characters are detected. Components of characters like *i* or *!* are completely grouped (see “Postleitzahl” on top of the image and “Post!” in the middle). Only in few cases, the procedure causes some trouble. For example the two character string “Ab” in the middle of the image of Figure 2 is not detected, because only text lines with more than three elements are extracted. Due to the resolution of 300 dpi and the small font size, some character segments are not correctly determined. To handle these cases, feedback from text recognizer is necessary. In Figure 11 there are some characters which look unusually (e.g. *ELRAD*). These characters are not collected to lines, because they are split into several small components. A line generation problem can be perceived in Figure 12. The lettering of the disk contains two incorrect text lines. This problem is caused by a small interline spacing where components of different lines have a smaller distance than the average intercharacter distance in the expected text lines.

In summary we can state that for the test documents the algorithm yields good results, that means all document parts satisfying the above mentioned requirements are extracted correctly. Only in some cases the algorithm produces incorrect or incomplete results. The following problems occur:

- Merged characters: If two characters build only one connected component, it is not split into two elements. If the number of merged characters is large[13], the corresponding text line is incomplete.
- Merged text and non-text pixel groups: If characters and graphics elements are merged, the resulting connected component is classified as non-text.
- Highly split characters: If characters are split into several components and they do not overlap

12. The test set mainly consists of 90 German business letters and 40 envelopes.
13. If more than 5 characters are merged, the corresponding component is often not picked up from the line.

each other, separate character segments are generated.

- Noisy background: If there are many small non-text components, typically generated from colored background, they are collected to characters, provided that they lie inside the line region.
- Short text string: If there are text strings consisting of less than three main components, they are not collected to lines.

After discussing the performance we evaluate the algorithm in terms of processing speed. Table 1 gives a breakdown of the processing times on a Sparcstation IPX for the entire analysis applied to three images.

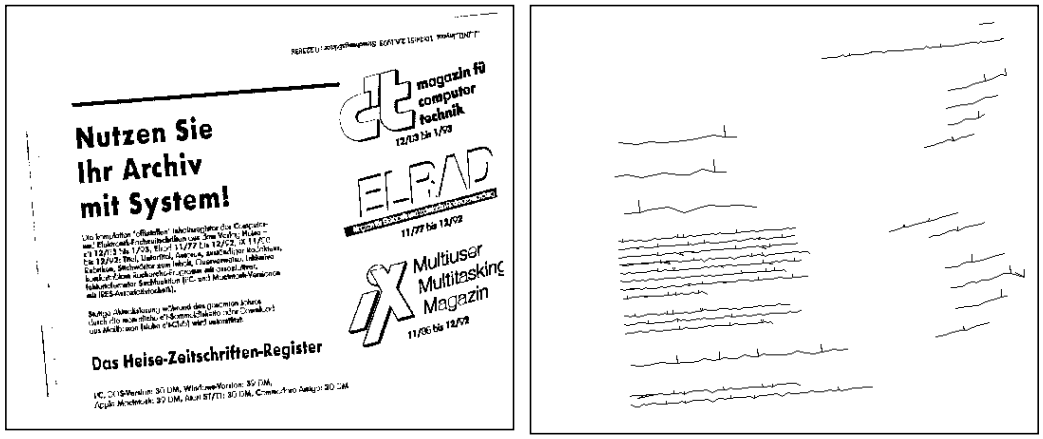

Figure 11: Example image No. 2 with corresponding text lines.

As shown in Table 1 the most time-intensive tasks are the temporary line generation and relaxation phases. They take more than 93% of the total runtime. This fact is mainly caused by the repeated calculation of text line features like regression line and expansion area. If a line is expanded or some elements are detached, it is necessary to re-calculate the features for obtaining a more adequate description.

	Figure 2	Figure 11	Figure 12
Number of Con.Components	1304	1770	1509
Con.Comp. Analysis	13,4 sec.	19,4 sec.	15,3 sec.
Filtering	< 0,1 sec.	0,3 sec.	< 0,1 sec.
Neighborhood Determination	1,5 sec	2,3 sec.	1,6 sec.
Temporary Line Generation	113,7 sec	143,7sec.	170,0 sec.
Relaxation	175,5 sec	384,2 sec.	327,2 sec.
Classification	< 0,1 sec.	< 0,1 sec.	< 0,1 sec.
Character Grouping	2,4 sec.	3,6 sec.	13,0 sec.
Word Grouping	< 0,1 sec.	10,0 sec	< 0,1 sec.
Non-Text Grouping	< 0,1 sec.	< 0,1 sec.	< 0,1 sec.
Total Runtime	306,9 sec.	563.,6 sec.	527,2 sec.

Table 1: Breakdown of CPU times for processing of three test images.

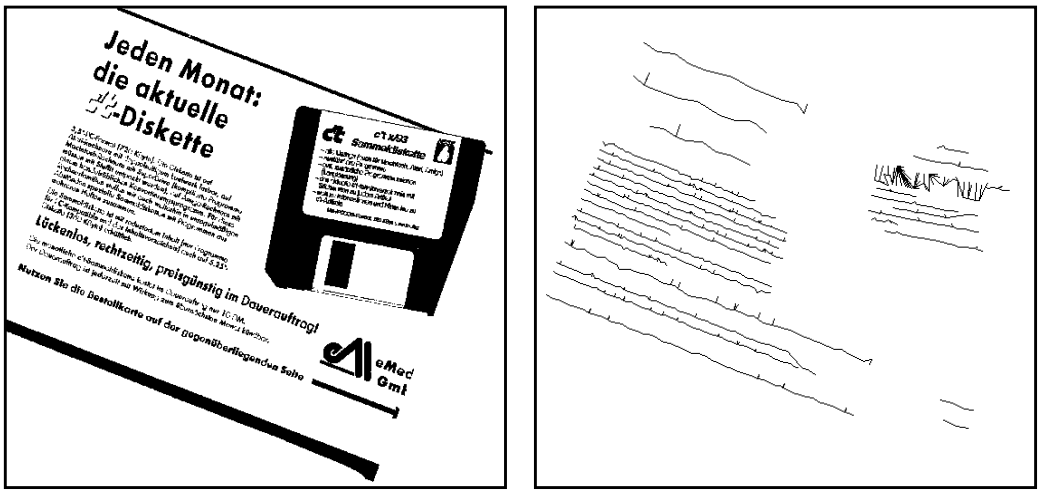

Figure 12: Example image No. 3 and detected lines. A defect can be perceived on the disk lettering which is caused by small interline spacing.

2.1.3 Summary and future work

A robust and effective algorithm for layout extraction of mixed-mode documents has been described. Black and white connected components represent the basic elements which are classified and segmented. As output, nontext components are collected in a separate block, whereas text elements are also grouped as lines, words and characters. There is no text recognition in the entire analysis. The classification only considers rough object features and relations to neighbor objects. The algorithm is highly reliable for documents that conform to a few constraints on their characteristics.

The approach presented has several advantages over other known techniques: it handles normal as well as inverse text, accepts characters in various font styles and sizes, is independent of text orientation, and allows the processing of characters that consist of more than one connected components.

The basis for the first point (handling inverse text) is established in the analysis of connected components where black and white components are detected. All subsequent phases process both kinds of text in the same way. The ability to process various font sizes is founded on the neighborhood determination. Instead of a pure resolution reduction or a smeared image in which the same neighborhood is generated for each object, we calculate specific neighborhoods depending on features of the components. Thus, we guarantee that, for characters of a large font as well as characters of small fonts, appropriate neighborhoods are established. Our temporary line generation, together with the relaxation phase, enable the processing of text in any orientation. The idea here is to find the longest homogenous line by detaching components from one line and reattaching them to another line. This phase strongly influences the quality of the entire system. Previous versions of our system indicated that without a priori knowledge about the text orientation, it was impossible to detect correctly all the text lines within documents like those illustrated in Figure 2, Figure 11 and Figure 12. Thus, the iteration of the relaxation process is a necessary and important phase. The ability to handle split characters is provided by the inverse filtering phase. In this phase the text line area is combed for components that are parts or fractions of characters.

The algorithm described has been tested on 150 documents and produces good results. Its performance depends on the quality of the connected components; that is, if many characters are merged or characters split into many small components, problems can occur. Similar problems exist

if the image contains too much noise. Generally, if the constraints mentioned are satisfied, all strings containing more than two characters are correctly and completely detected.

Our system can be influenced by some parameters. These parameters were initialized with standard values determined during implementation. Tests show that there is no need for manual adaptation of such parameters to specific input documents.

Future Work will concentrate on improving the character segmentation and on preprocessing images with too much background noise. Deciding whether a relative large connected component is one single character or two or more merged ones is a challenging task. A further problem occurs while segmenting split characters.

Images often contain a large amount of background noise complicating the task of document recognition and specially that of layout extraction. Reducing it saves a lot of computing effort and recognition impasses. Much of this image noise results from the binarization of grey scale images. Thus a promising approach to avoid such problems is to start the layout extraction with a grey scale image.

2.2 Font attribute identification

Today's text processing systems provide a vide variety of font styles. Each of them poses different problems to the recognition modules [Esakov 94]. For example, the shape of the same character can strongly differ between two fonts. This results in unwanted misclassifications. Furthermore, the segmentation quality is influenced by the input font, which results in character merges and split ups.

In general, each task of the text recognition problem is influenced by the characteristics of the input document, i. e. the font type, typical image distortions etc. It seems to be impossible to find general purpose solutions, which can handle all documents with equal high precision. Instead it seems to be more fruitful to build up specialized solutions for documents with different input properties.

The task is to identify properties which allow the division of the input documents into several classes. For each document class, specialized analysis algorithms have to be identified. To be able to employ the right specialists an automatic identification of the document class is necessary. It is obvious that faults in this decision can have a strong effect on the quality of the recognition results.

At *DAD* we plan to develop a sophisticated test module, which automatically analyses the recognition results and gives hints of the main difficulties for the analysis phases. Directed by the results of the test module a font attribute identifier will be developed, which computes features of the different parts of the input document. These features are subsequently used to initiate appropriate analysis algorithms. The font attribute identifier should evolve with ongoing project research. Each time new sources of errors are detected further input properties will be taken into consideration. Currently, the following font attributes are planed to be identified:

- Spacing: One discriminant characteristic of printed text is the character spacing. There are two main font classes, namely fixed-pitch fonts and mixed-pitch (proportional) fonts. While the intercharacter distances within fixed-pitch font is always of the same size, the character to character distances within mixed-pitch fonts vary strongly. Information about the character spacing assists the character segmentation process, where different characters can be merged or one character can be split into several parts.
- Serifs: Another possibility for grouping font attributes is to consider the shape of characters. There are two main classes: characters with serifs (serif fonts) and without serifs (sans serif fonts). Serifs are small cross-lines at the top or bottom of a character. Serifs are often responsible for character merges and strongly influence the width of a character's bounding box (e.g. i, i, l, l). Furthermore, characters of serif and sans serif fonts often have a different shape. Thus, different and specialized classifiers should be applied.

2.3 Character classification

The task of character classification is to recognize the character class of the segments containing binary picture information [Suen 86]. This is a typical classification problem. Today's character classification methods behave well on documents of good printing quality. However, even the best commercial devices barely reach 99% correct recognition when faced with poorly-printed or poorly-copied dense text. Further problems arise from the use of uncommon typefaces or small font sizes [Rice 94].

The topic of character classification can be divided into statistical, neural and syntactical/structural approaches [Schalkoff 92]. Especially statistical and neural approaches are very successful and are the commonly used techniques.

Statistical and neural methods for classification have many subtasks in common. Usually the input pattern is transformed into a feature vector which is subsequently mapped to a character class. It is important that the features discriminate the different classes with sufficient probability. Which features can be computed depends on the intuition and experience of the developer. Nevertheless, a classifier using too many features becomes slow and its recognition rate decreases because usually training sets are not large enough to exploit generalization capabilities of such a complex approach. Feature selection techniques are applied to identify subsets of good features [Kittler 86]. Alternatively feature extraction techniques can be applied, which transform a high dimensional feature space to a low dimensional one, preserving as much discriminating information of the original space as possible. All these methods can be used for statistical as well as neural approaches. The main difference is the subsequent classification method.

Statistical approaches use information about the distribution of the characters in the feature space to make their decisions. There is a wide variety of different approaches. Typical examples are Bayes-Classifier, Polynomial Classifier or Nearest Neighbor Classifier. Alternatively neural approaches use a net of neurons, each executing a simple mathematical function, to classify the characters.

At the current state of research the classification methods become more and more complex[14]. This is possible mainly because today's computer systems are powerful enough to execute the enormous number of computations in a reasonable time, which are necessary to train these classifiers. Complex classifiers require very large training sets which also become available today, especially by the use of distortion models, allowing the efficient generation of synthetic training samples.

Syntactic approaches express each pattern, i. e. a pixel image of a character, as a composition of its components, called subpatterns or pattern primitives [Fu 86]. An analogy is drawn between the structure of patterns and the syntax of a language. The recognition of a pattern is usually made by parsing the pattern structure according to a set of syntax rules. Though syntactic methods for character recognition are widely used, they seem to be less successful for this task than statistical methods.

In *DAD* we are not working actively in the research area of character classification. We only want to apply successful techniques and experiences of other researchers in this field. For example, we will realize a polynomial and a neural network based approach for character recognition. Nevertheless, we are interested in building classifiers for discrimination between character classes which impose difficulties on conventional systems, i. e. commercial OCR systems.

As our basis for character recognition we integrate commercial OCR systems into our text recognition module. It should be mentioned that commercial OCR systems are not only character classifiers, but more complete devices, which are also able to do preprocessing and layout extraction. This imposes problems on the integration into our system architecture, which have to be resolved.

---

14. using high dimensional polynomials or several hidden layers each consisting of many neurons

Thus, in *DAD* we will develop an interface to control all OCR devices. We use the results provided by our segmentation to define the appropriate image regions for the recognition task.

Because alternative systems commit different errors we will use several classifiers (commercial devices as well as our own character recognizer) to reduce the error rate by combining their results (see Section 2.4). So far we have incorporated two commercial products, namely Socrates and ScanWorX. Both systems get binary images as input and produce characters as result. The different output formats are transformed into a homogenous representation. In order to fulfill this task parsing mechanisms for the various proprietary formats are necessary. Of special interest is a correct interpretation of the information attached to recognized characters as there are marks and belief measures.

Ongoing research will focus on the combination of our segmentation with the zoning information of the commercial products.

2.4 Voting of OCR results

So far, we have described independent OCR systems all working in parallel. Since none of the OCR devices offers a recognition accuracy which is good enough for all documents with respect to the demands of the text analysis, we have developed a so called “voting machine” to combine the individual results of all classifiers. The voting machine should produce a combined output, which is more reliable than any of the individual results. To do so, voting focuses on the individual classifier’s strength and avoids their weaknesses. Another aspect of voting is the homogenization of the character recognition results. Furthermore, voting performs some kind of information reduction, suppressing unimportant results, which prevents a combinatorial explosion of input data for subsequent analysis phases.

Since we have integrated commercial OCR devices, our voting component has to impose only little restrictions on the classifiers. The remainder of this subsection will discuss some fundamental voting aspects and will present our proposed system architecture together with first results achieved.

2.4.1 Basics

Although current OCR devices produce high reliable recognition results for good quality documents, they reveal specific weaknesses on lower quality documents. The recognition accuracy of commercial devices is documented in [UNLV 94]. Here one can observe that even the best devices barely reach 100% accuracy. The character accuracy for the DOE[15] sample ranges from 95.5% up to 98.4% with an average of 97.2%. The progress made in recognition technology is documented separately. From [UNLV 93], one can see, that the average accuracy of the same devices was 96.4%. Bradford and Nartker stated in 91 [Bradford 91], that “most applications, however, require higher database accuracies, typically on the order of 99.8%.” This also holds for our application scenario, where we focus on a subsequent text analysis phase.

To yield best possible word recognition results, which are of much more interest for any kind of text analysis than character recognitions, we strongly depend on reliable character recognition results. In order to enable a fully automatic processing of recognized text, further improvements in recognition accuracy are essential. One promising approach is the combination of different classifiers. Franke and Mandler [Franke 92] stated, that “no known approach is powerful enough to solve the problem in general. However, when applying different classifiers to the same recognition task, it turns out that they don’t make the same errors. Having optimal decision procedures to combine the output of different classifiers working in parallel on the same entity would improve the overall per-

15. U.S. **D**epartment **O**f **E**nergy

formance remarkably.”

In recent years, voting algorithms and techniques have evolved and have gotten more sophisticated. The principles of combination, of result representation, and the use of probabilistic reasoning are increasingly refined. But these advances do not affect the global architecture of a system for classifier combination as shown in Figure 13. Generally, the input of voting is the output directly

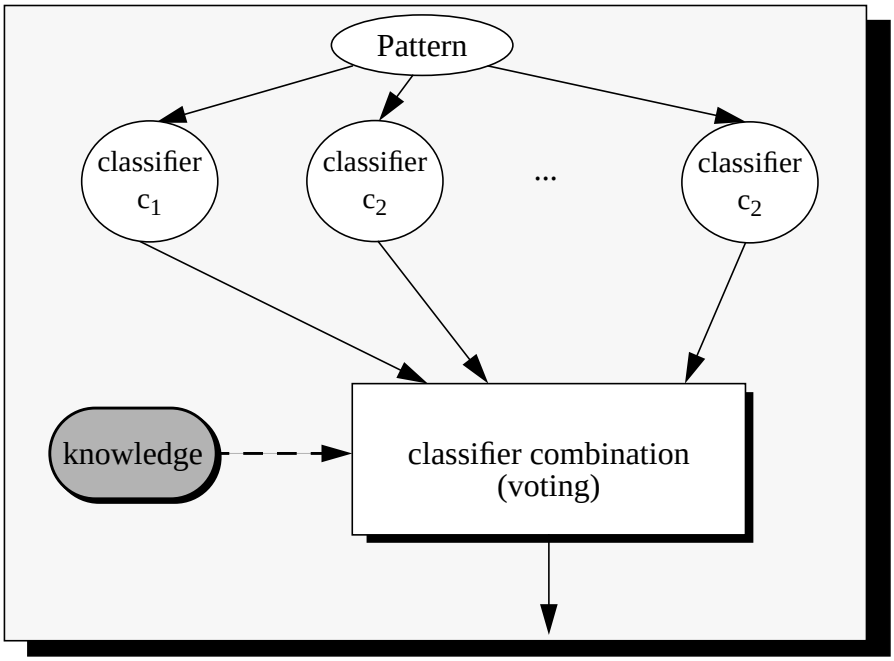

Figure 13: The global architecture of a classifier combination system is shown. The knowledge base is optional.

produced by the individual classifiers which is combined and sometimes supported by conflict resolution knowledge.

In the remainder of this section, we discuss some issues which are fundamental for classifier combination problems but not limited to OCR or text recognition in general. From our point of view, there are four main aspects to be worked out:

1. degree of integrity
2. representation of classifier results
3. combination of classifier results
4. methodological and representational restrictions

Before describing these aspects in greater detail, we will introduce a more formal notation suitable for most sorts of recognition problems (cf. [Huang 93]):

Let P be a pattern space consisting of M mutually exclusive sets $P = \bigcup_i C_i$ with each C_i, $\forall i \in \Lambda = \{1, 2, \ldots, M\}$ representing a set of specified patterns called a class. For a sample x from P, the task of a classifier e is to assign x one index $j \in \Lambda \cup \{M + 1\}$ as a label. If $j = M + 1$, x is rejected by classifier e, which means, that e has no idea about the class of x, otherwise, x is assumed to belong to class C_j. Thus, a classifier can be regarded as a function box receiving an input sample x and producing a label j, shortly denoted by $e(x) = j$. In practice, classifiers may enrich their output by additional confidence values or less trusted hypotheses. These aspects will be discussed within the representational alternatives.

2.4.1.1 Degree of integrity

The degree of integrity serves as a measure how strongly the individual classifiers are directed by the voting algorithm. A high degree of integrity denotes that the voting machine directs all activations of the classifiers. It "knows" about strengths and weaknesses and runs only those classifiers which promise reliable results depending on the current situation. If the degree of integrity is low, the voting component has no effect on the basic classifiers and considers their output only.

2.4.1.2 Representation of classifier results

The second aspect concerns the representation of classifier results. Here, three alternatives are possible (cf. [Xu 92]):

1. ***abstract level***: Each classifier e provides a single result/label j without any further information, i.e. $e(x) = j$, $j \in \Lambda \cup \{M+1\}$.
2. **rank level**: Each classifier provides an ordered list of ranked results, where the first element is the most likely one, i.e. $e(x) = L$, where $L \subseteq \Lambda$ is an **ordered** list. A reject is denoted by $L = \varnothing$
3. **measurement level***:* Each classifier produces alternatives along with a real value indicating the recognition confidence. The values have not necessarily to be taken from the interval [0,1] indicating a probability, but also can be distance measures to given reference patterns. In the latter case the values come from the interval [0,∞] and lower values indicate lower distance and thus higher confidence. Formally: $e(x) = B_e$, where $B_e = \{(i_1, b(i_1)), \ldots, (i_n, b(i_n))\}$, $\{i_1, \ldots, i_n\} \subseteq \Lambda$, $n \leq M$, and $b(i_s)$ denotes some kind of degree that e considers that x has label i_s. Rejection is denoted by $B_e = \varnothing$.

Obviously, the amount of data increases from alternative (1) to alternative (3). If the voting component has no influence on the result representation, it has to transform all results in a consistent way. This comes along with minor difficulties when decreasing the level of detail, since results from the rank or measurement level can be directly transformed into the abstract level by omitting all but the best result. Transformations from less detailed levels into more detailed ones are more difficult to perform, especially those resulting in the measurement level.

2.4.1.3 Combination of classifier results

The representation selected strongly influences the underlying technique for combining the individual results but also the production of the final output. It's impossible to combine results of individual classifiers represented differently without additional efforts. Thus, we assume that all results are represented on the same level. The output of combination may be represented on a different level, but in most cases it will be the same.

As abovementioned, the classifiers' output on the **abstract level** is a single class decision j or a reject. Given K individual classifiers and an input sample x, each classifier assigns a class label j_k to x, where $k \in \{1, \ldots, K\}$. Representing the vote output on the abstract level too, the problem of combination is to assign x a unique class label $E(x) = j$, where $j \in \Lambda \cup \{M+1\}$. For convenience, the classifier's output can be represented as binary characteristic function (cf. [Xu 92]):

$$T_k(x \in C_i) = \begin{cases} 1, \text{ when } e_k(x) = i \text{ and } i \in \Lambda \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

One widely known method for combination, the „majority voting", can be expressed with the following formula:

$$E(x) = \begin{cases} j, \text{ when } T_E(x \in C_j) = max_{i \in \Lambda} T_E(x \in C_i) \\ M+1, \text{ otherwise} \end{cases} \text{, where} \quad (2)$$

$$T_E(x \in C_i) = \sum_{k=1}^{K} T_k(x \in C_i)\ ,\ i = 1, \ldots, M \quad (3)$$

Thus, the majority vote assigns x the class label which is most often presented from the individual classifiers. A more general combination rule is presented in [Xu 92]. Various modifications can yield more or less conservative decision rules. We will not discuss these modifications within this report.

For the **rank level**, a thorough overview of combination strategies is provided by Ho [Ho 92a]. Deviating from the more general concept of rankings as presented in Section 2.4.1.2, Ho assumes, that all given classes are ranked by each individual classifier. Higher rank positions[16] of a class denote higher trust in this class. Two principal methods are discussed: class set reduction and class set reordering.

The goal of the reduction method is to extract a subset of classes which hopefully captures the correct class. In class set reorderings, the objective is to derive a combined ranking of the given classes, such that the true class is ranked as close to the top as possible.

For class set reduction, the criterion for success is two-fold: The size of the result set should be minimized, and the probability of inclusion of the true class in the result set should be maximized. Two approaches are proposed in [Ho 94] to reduce the class set:

1. The intersection approach

   In the intersection approach, a large neighborhood is obtained from the top of the class ranking by each classifier. The intersection of these neighborhoods is the result set. Thus, a class is in the result set if and only if it is included in *all* the neighborhoods. The sizes of the neighborhood can be determined by the ranks of the true classes in the worst case. Therefore, in a training phase thresholds for each classifier are calculated, denoting the worst rank position of a correct class for the respective classifier.

2. The union approach

   In the union approach, a small neighborhood is obtained from each classifier. The union of these neighborhoods is output as the result set. Thus, a class is in the result set if and only if it is in *at least* one of the neighborhoods. Like for the intersection approach, thresholds can be calculated for determining the individual neighborhood sizes.

In both approaches, thresholds are chosen which ensure not to lose the correct class decision. To further reduce the size of the result set, thresholds can be calculated by approximative methods (cf [Ho 92a]).

For class set reordering, the criterion for success is the rank position of the correct class: It should be ranked as near to the top as possible. In other words, the probability of having the true class near the top of the combined ranking is higher than those in each of the original rankings. In [Ho 92b], three methods for class set reordering are presented:

1. The highest rank method

   For an input pattern x and K classifiers, each class receives K ranks. Now, a score is assigned to each class reflecting the highest rank position of this class in any of the individual rankings. All classes are ordered according to their new score values yielding the combined ranking. One

16. The highest rank position is position 1.

disadvantage of this method is that the combined ranking may have many ties, i.e. classes are ranked to the same position. Therefore, this method is only useful if K is small relative to the number of classes M.

2. The Borda Count method

   In the context of group decision theory, classifier combination can be viewed as committee voting problem. Here, the mapping from a set of individual rankings to a combined ranking is referred to as a *group consensus function*. One of the more popular group consensus functions is the so called *Borda Count* [Black 63]: For any class C_i, $i \in \Lambda$, let $B_k(C_i)$ be the number of classes in P which are ranked below C_i by classifier e_k in its individual ranking L_k. The Borda Count for class C_i is

$$B(C_i) = \sum_{k=1}^{K} B_k(C_i) \tag{4}$$

   The combined ranking is given by arranging the classes according to their Borda Counts in descending order. Many variants of the Borda Count exist, including one which is able to handle ties. In general, the problem of ties is not solved but appears less often than in the Highest Rank method. One problem of this method is the underlying assumption of classifier independence. Advantages of the Borda Count method are that it is easy to implement and requires no training. The main disadvantage is that all classifiers are treated equally, independent of their actual performance. The next method overcomes this shortcoming.

3. The logistic regression method

   Simply spoken, this method is a variant of the Borda Count where each classifier is assigned a specific weight, expressing its overall performance. Thus, more trusted classifiers get higher weights and have more influence on the combined decision than less trusted ones. The weights are estimated by logistic regression using training sets (cf. [Ho 92a],[Ho 92b]).
   Again, each class is assigned a score and all classes are reranked according to their scores in descending order. The Logistic Regression method is the most sophisticated method among the three presented methods. It's able to focus on the individual strengths of each classifier. Its main disadvantage is the expensive parameter estimation, which requires large training sets.

The **measurement level** provides class decisions along with confidence values for these classes. The confidence values can be similarity as well as distance measures. They need not be taken from the interval $[0, 1]$ expressing some kind of probability. One problem is the transformation of different confidence measures utilized by different classifiers. Another problem is the combination of the class confidence values to yield a combined confidence value for each class. The latter problem will be sketched first, so let's assume all classifiers make use of the same confidence measure. Since the confidence values can be treated as vague information, all kinds of probabilistic reasoning can be employed for their combination. The two most popular methods are those based on Bayes' Theorem and Dempster/Shafer Formalism.

The Bayesian Probability is strongly related to the "classic" probability theory based on the so called Kolmogorov Axioms:

Let Ω be a finite set of independent events[17]. A function $P: 2^{\Omega} \rightarrow [0, 1]$ is called a probability function, if and only if it satisfies the following conditions:

- $P(\varnothing) = 0$
- $P(\Omega) = 1$

17. Sometimes, Ω is called *universe*.

- $P(A \cup B) = P(A) + P(B)$, for every $A, B \subseteq \Omega$ where $A \cap B = \varnothing$

Bayes' Theorem utilizes the notion of conditional probability. Assuming classifier independence, the following equality holds after transformation:

$$P(i) = P(x \in C_i) \times \frac{\left(\prod_{k=1}^{K} P(x \in C_i | e_k(x) = B_e(k)) \right)}{\prod_{k=1}^{K} P(x \in C_i)} \tag{5}$$

The conditional probabilities $P(x \in C_i | e_k(x) = B_e(k))$ can be estimated easily (see [Xu 92]). Xu et al. also propose to approximate (5), if the a-priori probabilities $P(x \in C_i)$ are unknown:

$$P(i) = \prod_{k=1}^{K} P(x \in C_i | e_k(x) = j_k) \times \eta \text{, where} \tag{6}$$

η is a constant, which ensures that $\sum_{i=1}^{M} P(i) = 1$. The final class decision for sample x is now:

$$E(x) = \begin{cases} j, \text{ when } P(j) = max_{i \in \Lambda} P(i) \\ M+1, \text{ otherwise} \end{cases} \tag{7}$$

One of the major disadvantages of the Bayesian probability is its inability to express lack of knowledge, which is a result of the fundamental condition $P(A) + P(\bar{A}) = 1$. Different approaches have been proposed to overcome this inability, among which the notion of belief or fuzzy measure plays a central role. A special case of fuzzy measure is the more popular Dempster/Shafer theory (see [Shafer 76],[Tahani 90]). Omitting any interpretation, the difference between D/S theory of evidence and the classic probability axioms is expressed in the following basic conditions for belief functions:

- $bel(\varnothing) = 0$
- $bel(\Omega) = 1$
- $bel(A \cup B) \geq bel(A) + bel(B)$, for every $A, B \subseteq \Omega$ where $A \cap B = \varnothing$

Only the third axiom differs in its choice of relational operator. One important concept for defining belief measures is based on the notion of basic probability assignments:

Let Ω be a universe. A function $m: 2^{\Omega} \rightarrow [0, 1]$ is called basic probability assignment (bpa), if and only if the following conditions are fulfilled:

- $m(\varnothing) = 0$
- $\sum_{A \subseteq \Omega} m(A) = 1$

Now, assuming $\Omega = \{A_1, \ldots, A_M\}$, the sets $\{A_i\}$, $\forall (i \in \Lambda)$ are only one part of the elements of 2^{Ω}. Thus, it is possible to have $\sum_{i=1}^{M} m(A_i) < 1$, which implies $m(A_i) + m\left(\bar{A}_i\right) < 1$. Having

assigned the bpas, a belief function is induced in the usual way (cf. [Shafer 76]), where $bel(A) = \sum_{B \subseteq A} m(B)$.

In our classification scenario, the M complete and independent alternatives are given by $A_i = x \in C_i, \forall (i \in \Lambda)$. The K classifiers $e_1, \ldots, e_K$ output for a given sample x K evidences $e_k(x) = B_e(k)$. Using these evidences, the bpas for all classifiers have to defined. Afterwards, the bpas can be combined using Dempster's rule of combination, yielding a new bpa for the combined evidence: $m = m_1 \oplus m_2 \oplus \ldots \oplus m_K$. Since this rule is associative and commutative, it can be applied iteratively (cf. [Shafer 76]).

2.4.1.4 Methodological and representational restrictions

Another important aspect of combination are the restrictions concerning the methodology used by the individual classifiers to produce results as well as the representation of these results. Neglecting the technical aspects, a more "tolerant" voting component would combine results of classifiers using different classification approaches and utilizing different representation levels, while a more restrictive voting component requires, e.g., a homogenous representation or even a specific classification algorithm.

2.4.2 State of the art

Having discussed the basics of classifier combination in the previous section, we will now present voting techniques currently proposed by different researchers. The various techniques or complete systems will be classified according to the four main aspects, degree of integrity, representation of results, combination of results, and restrictions.

1. Ho, resp. Ho et al. (see [Ho 92a],[Ho 92b],[Ho 94]), have developed a classifier combination system for the task of word recognition. The combined decisions should be better than those given by the single classifiers applied in isolation. The decision combination function "should take advantage of the strengths of the individual classifiers, and avoid their weaknesses." (see [Ho 92a] p. 39). The approach could be characterized as follows:
   - integrity: Classifiers could be selected dynamically at classification time. Thus, the overall performance could be enhanced by selecting only those classifiers promising reliable results depending on the given situation.
   - result representation: The individual classifiers output a ranking: $e_k(x) = L_k$. Ho assumes, that all classes are ranked with respect to the given sample x.
   - combination: Different combination methods are developed and their precision and recall values are compared. All methods belong to the class set reordering approach, which namely are: Highest rank, Borda count, and Logistic regression.
   - restrictions: There is only one restriction for the individual classifiers: Their output has to be ranking of all classes. No other restrictions are imposed on the classifiers.

   Three completely different methods for word recognition were combined in the tests, yielding a 9.2% improvement in recall compared to the best individual classifier.
2. Mandler and Schürmann [Marzal 93] combine the output of character classifiers:
   - integrity: The voting component has no effect on the individual classifiers. It combines only their output.
   - result representation: The results are represented on the measurement level. All measures

have to be distance values to given reference patterns.

- combination: Dempster/Shafer theory of evidence
- restrictions: The voting component imposes high restrictions on the individual classifiers. All classifiers have to be nearest neighbor classifiers, ensuring their confidence value to be a distance measure.

Franke and Mandler [Franke 92] showed the superiority of their classifier combination approach over single classifiers. They also compared the combination schemes based on Dempster/Shafer and Bayes and concluded, that both approaches do no differ significantly.

3. Huang and Suen [Huang 93] discuss the influence of the chosen representation on the overall performance:
   - integrity: The voting component has no effect on the individual classifiers. It combines only their output.
   - result representation: The results are represented on the measurement level. The measures could be similarity-, or distance measures, or even subjective confidence values.
   - combination: The combination is performed with the Linear Confidence Aggregation Method (LCA). In a first step, the different measures are transformed into one single measure. These measures are added for each class, yielding a confidence value for this class. The class with the highest confidence value greater than a given threshold will be output.
   - restrictions: The voting component imposes only one restriction on the individual classifiers: their output has to be on the measurement level. The measures are free of choice

   Although, for their experiment only two polynomial classifiers were integrated, Huang and Suen showed the better performance of their LCA method compared to a majority voting or Bayes based voting.
4. Rice et al. [UNLV 94] presented the results of a test of two voting systems in their 1994 report. The system developed at ISRI will be discussed shortly:
   - integrity: The voting component has no effect on the individual classifiers. The latest versions submitted by Caere, Calera, ExperVision, Recognita and XIS are used for combination.
   - result representation: ISRI deviates from the given scheme. Here, only strings, or more precisely substrings, are used for voting. The voting component has only to process those regions, where output differs.
   - combination: majority vote
   - restrictions: No restrictions are imposed on the classifiers. Their output is transformed into strings, all additional information such as less trusted hypotheses or confidence values will be neglected.

   Rice et al. stated, that “in general, the voting systems perform best when the text obtained from the OCR systems is of high accuracy. ... if none, or perhaps only one, of the OCR systems is able to read a given text region, a voting system has little or no chance of producing accurate output.” The overall error reduction rate yielded by their voting machine was 42%.
5. Bartell et al. [Bartell 94] propose a classifier combination procedure to combine expert/classifier opinions in the domain of information retrieval.
   - integrity: The voting component has no effect on the individual classifiers.
   - result representation: The classifiers provide results on the measurement level. These results are documents together with a numerical estimate of the relevance of documents to a query.

- combination: The combination is based on a linear combination model. Model parameters are optimized globally, thus, the classifiers need not be independently.
- restrictions: The voting component imposes no restrictions on the classifiers, they only have to provide relevance measures for each document.

In two different tests, Bartell et al. yielded 12% and 47% performance improvement compared to the best individual classifier. They stated, that an improvement is possible, even if only two classifiers are combined, one of which performs relatively poor compared to the other.

The abovementioned approaches are chosen exemplarily, to sketch some of the current work done in the classifier combination field. Other approaches are known as well: [Chen 93], [Ling 89], [Lopresti 94], [Plessis 93], [Sabourin 93]. For a short discussion see also [Ho 92a] pp. 9-15.

The topics of classifier combination also have strong relations to the field of decision fusion (see [Dasarathy 94]).

2.4.3 Proposed solutions and system architecture

In this section we will present our approach for classifier combination and will discuss the proposed system architecture. First of all, we will not develop a single voting mechanism. Instead, we will compare different approaches for result representation and classifier combination. To improve voting performance, conflict resolution knowledge should be integrated consistently into the combination process. Thus, automatic learning of a classifier's strengths and weaknesses should take place to contribute to high reliable recognition results.

Basically, the proposed system architecture is two-fold, it consists of an off-line learning component and the classification component itself. The learning component has the objective to gather conflict resolution knowledge to support the classification component. The combination takes place in the classification component where the recognition results of the individual classifiers are processed and a combined result is output. To resolve arising conflicts, e.g. when the individual classifiers don't vote for the same class, the conflict resolution knowledge is utilized to yield a single class decision whenever possible. Figure 14 shows the proposed system architecture in detail.

In the **learning component**, confusion matrices for all character confusions of the individual classifiers should be computed and stored in the knowledge base. Depending on a classifier's ability to output less trusted hypotheses or hypotheses together with confidence values, the confusion matrices should be more or less detailed. For our approach, it is not sufficient to keep track with the common confusions only. Instead, we also need the probabilities for each confusion (or the correct result respectively) to transform the different individual measures into a single measure. Different kinds of confusion matrices CMi are currently under consideration:

- CM1 traces only the confusions of the top candidate of each classifier, neglecting other candidates and confidence measures
- CM2 traces the confusions made for each candidate, depending on its position. Again, the confidence values will be neglected. Thus, one can think of one confusion matrix for each position: A CM for the top candidate for each classifier, a CM for the second trusted candidate etc.
- CM3 traces the confusions made for each candidate depending on its position and confidence measure. To restrict computation time and space, this CM will be intermixed with CM2, such that the confidence values are considered for the top choice only. All other candidates will be treated position dependent.

The **classification component** performs the actual combination. The active components are: adjustment component, combination component, result estimation component, knowledge incorporation

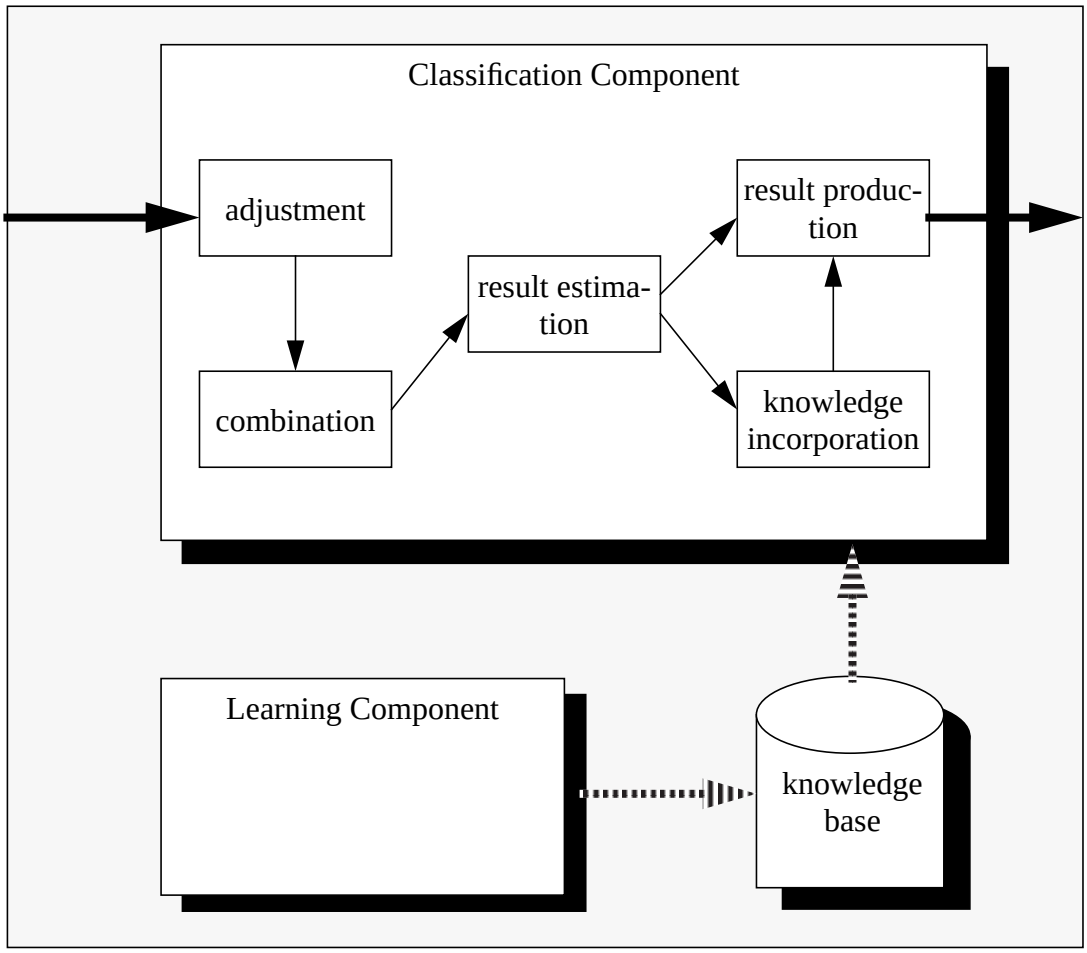

Figure 14: Global system architecture for the voting component. Normal arrows denote flow of data, dashed arrows denote flow of knowledge.

component, and result production component (see Figure 14). The activation of the knowledge incorporation component depends on the result estimation component, all other components are activated always.

First, all individual classifier results are transformed onto the same level of representation (adjustment). Thus, the *adjustment component* should be easily configured to represent results on the abstract, rank, or measurement level. For the abstract or rank level this comes along with minor difficulties. For the measurement level, all confidence values have to be transformed into one single measure. This can be achieved by utilizing the confusion matrices' entries.

The *combination component* performs the actual result combination. As described above, we will compare different representation and combination strategies. As a special constraint, the global output has to be represented on the measurement level. Thus, we have to adapt some of the combination schemes to provide additional confidence values.

For the abstract level, a simple majority vote will be developed. Here, the output is no longer a single class decision. Rather, a number of classes together with their $T_E(x \in C_i)$ values (see Section 2.4.1.3) are output.

Combining results on the rank level is performed by a modified version of the Borda count (see Section 2.4.1.3). This modification is necessary to contribute to the distinct length of the individual classifier rankings. The aforementioned Borda count performs well only on equal length rankings, so we use a slightly different definition:

Let n_k denote the length of classifier's e_k ranking, $k = 1, \ldots, K$. Define $r = max_k(n_k)$. Now, for each class C_i, $i \in \Lambda$, let $B'_k(C_i)$ denote the number of classes ranked **above** C_i in the ranking of classifier e_k. $B'_k(C_i) = r$, if C_i does not appear in e_k's ranking. The modified Borda count can be defined as:

$$B'(C_i) = \sum_{k=1}^{K} (r - B'_k(C_i)) \tag{8}$$

To contribute to the overall classifier performance, a weighted version of this modified Borda count will be used:

$$B'_W(C_i) = \sum_{k=1}^{K} w_k \times (r - B'_k(C_i)), \tag{9}$$

where w_k are classifier precisions learned during off-line learning. The output is the ranking obtained together with the $B'_W(C_i)$ values for every class C_i in the ranking.

The combination on the measurement level will be performed by Dempster/Shafer's theory of evidence. Remember that classifier e_k's output for given sample x can be represented as $B_k(x) = \{(i_1, b(i_1)), \ldots, (i_n, b(i_n))\}$, where $\{i_1, \ldots, i_n\} \subseteq \Lambda$ and $n \leq M$.

Since all confidence values are transformed into a single measure expressing the probability for a given class of being the correct one, the basic probability assignments can be made directly:

- $m_k(A_{i_s} = x \in C_i) = b_k(i_s)$, where $i_s \in \Lambda$ and $b_k(i_s) \neq 0$
- $m_k(\Omega - A) = 1 - \sum_{i=1}^{M} m_k(A_i)$, where $A = \bigcup_{i \in \Lambda} A_i$ with $m_k(A_i) > 0$

The actual combination is performed by Dempster's rule of combination and a final basic probability assignment is obtained which directly induces a belief function in the usual way.

The *result estimation component* tries to estimate the quality of the intermediate combination result. Utilizing a simple heuristic, the momentary results are categorized into high reliable or less reliable. Always, the quotient of the confidence values of the top and second candidate are compared to some empirically determined threshold δ.

The *knowledge incorporation component* has the objective to improve the intermediate results categorized as less reliable. Thus, it employs more knowledge from the knowledge base than already used in the adjustment component.

For the abstract level, the confusion matrix is utilized to attain the most probable class for a recognition result. This class is then used for further processing.

For the rank level, more sophisticated techniques than the weighted modified Borda count will be employed. To contribute not only to the overall classifier performance, but to its specific strengths and weaknesses, the global weight will be substituted by a local weight, yielding:

$$B'_{CM}(C_i) = \sum_{k=1}^{K} cm_k(j_k, i) \times (r - B'_k(C_i)), \tag{10}$$

where $cm(j_k, i)$ is the frequency of C_i being the correct class in the case of C_{j_k} being classifier's e_k top choice.

As for the rank level, a different confusion matrix is utilized for the measurement level, too. The actual combination remains unaltered. The changes solely effect the definition of the basic probability assignments, where the entries of CM3 are utilized now.

Task of the *result production component* is the generation of results, which have to be proper integrated into the whole layout architecture. Since all results have to be represented on the measurement level, these measures have to be calculated for those results represented on the abstract or rank level respectively. For the measurement level, the measure of belief is directly used as final confidence value. As mentioned before, confidence measures are attached to all class hypotheses on the abstract or rank level. These measures are normalized by the number of classifiers who took part in the voting process. The normalized measures serve as confidence values for the final output.

2.4.4 Test results

In the last research period, a first prototype was developed. This prototype already incorporates most of the abovementioned concepts. Its modular and object oriented design supports further improvements, other combination principles as well as confusion matrices can be easily integrated.

First results of the voting system and comparison to the individual classifiers contributing to the voting are shown in Table 2:

Recognizer	Recall [%]	Precision [%]
C_1	99.4	99.4
C_2	97.6	98.0
C_3	83.9	91.7
C_4	72.7	76.5
V_ABS	**99.5**	**99.5**
V_RANK	**99.5**	**99.5**
V_MEA	**99.6**	**99.7**

Table 2: Overall performance of individual classifiers and different voting strategies. The names for the different voting strategies directly correspond to the chosen result representation, namely abstract, rank, or measurement level.

Recall and precision are defined with their usual meanings:

$$Recall = \frac{c}{n}, Precision = \frac{c}{n-r}, \tag{11}$$

where c is the number of correct classified samples, n is the total number of samples, and r is the number of rejected samples.

In this table, both, C_1 and C_2, are commercial OCR devices which are integrated homogeneously into our text recognition system. C_1 outperforms C_2 on our german documents, because C_2 is unable to deal with *german umlauts*. C_3 and C_4 are character recognizer developed in our predecessor project ALV. 78 documents were included in the current test samples, which were the same documents used for the construction of the confusion matrices. Due to character missegmentations, only one third of the characters, in total 26584, take part in the voting process, the others are neglected. For all characters which are correctly segmented, ground truth text is provided and com-

pared to the classification results. Only the top candidates were compared to the ground truth. When a classifier produced more than one top candidate, the result was correct, if one of the top candidates was correct.

From the table we can see, that C_1 performs best among the individual classifiers. Although a high recognition rate is yielded by a single classifier and lower ones by all others, especially by C_3 and C_4, the voting component is capable of improving recall as well as precision. Best results are obtained by the voting with measurement level representation. Here, 41% improvement in recall and 44% improvement in precision are attained.

Currently, the evaluation of the different voting strategies is ongoing and we will be able to present more detailed results in the next research period. Nevertheless, the above results are encouraging and justify the proposed voting approach.

2.4.5 Summary and future work

In the previous sections, we have presented a suitable system architecture for a voting machine and have demonstrated first results of different voting strategies obtained. Since it is our objective to compare different voting strategies, we had special emphasis on a modular and easy to adapt system design. In Section 2.4.1, we have presented a scheme for classifying current voting approaches. With respect to this scheme, our proposed approach could be characterized:

- integrity: The voting component has no effect on the individual classifiers. In our case, this is a necessity, since we employ commercial OCR devices to improve recognition accuracy. Thus, only the output is processed
- result representation: Currently, the classifiers have to represent their output on the measurement level. A transformation (adjustment) is performed as a first voting step, so that results can be combined on any of the three known levels.
- combination: Different combination methods are employed, depending on the chosen representation. For the abstract level, a simple majority vote is employed, for the rank level a version of the weighted Borda count is employed, and for the measurement level Dempster/ Shafer's theory of evidence is employed. Each combination method is supported by conflict resolution knowledge, namely various kinds of confusion matrices. Thus, a normalization of confidence values and actual probabilities can be obtained.
- restrictions: The voting component imposes only little restrictions on the individual classifiers. They have to represent their results on the measurement level. This constraint can be weakened without greater effort, since the adjustment component transforms all results on the chosen representation level.

In the last research period, we payed special attention to the consistent transformation of different confidence values obtained by different classifiers. Thus, we stored confusion matrices calculated in an off line learning into a knowledge base. These matrices can be used to transform confidence values and to improve recognition results. Encouraging results could be documented, especially for the voting approach based on Dempster/Shafer's theory of evidence, where a 41% recall and 44% precision improvement were achieved.

Future work will concentrate on further evaluation of voting results compared to individual classifiers. Strengths and weaknesses will be elaborated, with special emphasis on the comparison of the voting component to the best individual classifier.

One challenge directly concerning the voting component is the handling of different character segmentations and missegmentations. Here, concepts evolve to represent those results in a directed graph structure and to preprocess this structure before the actual voting takes place.

For the voting itself, approaches based on neural nets or polynomial classifiers are under consideration. Here, the individual classifier's output will be treated like "normal" feature vectors, and known training strategies can be employed.

Furthermore, having evaluated the whole text recognition module, we will better reconcile the voting component with the contextual postprocessing component.

2.5 Lexical postprocessing

Even a human being is unable to recognize the input characters of a document with sufficient accuracy if he does not use additional contextual information to resolve this classification problem [Toussaint 78]. This is especially true if the input image is of bad quality. One important source of information is the vocabulary, which defines the allowed words of the document. Much research has taken place and several solutions have been proposed to integrate this knowledge into the overall recognition process of text input by a computer system ([Hall 80], [Hull 88], [Sinha 93]).

The solutions for integrating lexical knowledge can be divided into two different approaches. The first one uses incomplete knowledge about the words of the vocabulary ([Hull 88], [Riseman 74], [Shinghal 79]). Binary or probability based character transitions are employed to verify the output of the character recognition stages. Those techniques are very fast but have only a low recognition accuracy. Furthermore they can also produce character strings as output, which are not correct words, because of their incomplete knowledge about the vocabulary.

The second approach to integrate lexical knowledge uses a dictionary containing all known or allowed words [Srihari 1983]. The *character hypotheses lattices*[18], resulting from the character recognition phase are compared with the words of this dictionary using some kind of distance measure. There exists a wide variety of different distance measures, some based on the concept of accumulated editing costs some on probabilistic models for the distortions which take place during recognition ([Kashyap 84], [Weigel 94]). Techniques using this approach do have a good recognition accuracy but they tend to be slow especially if a large vocabulary is used. In this case a character hypotheses lattice has to be compared with many different words slowing down the overall processing speed. To overcome this problem several search strategies have been proposed which speed up the approach by using efficient data structures as well as heuristics to prevent an exhaustive search over the whole vocabulary.

At the current state of research at *DAD* we have focused our research interests on lexical postprocessing methods using complete knowledge about the allowed or used words. This has mainly one reason. The text analysis phase of a document analysis system needs high recognition accuracy on the word level, which can best be achieved by such methods. If an unknown word is input[19], the best the system could do, is to signal this to the other phases. Even if it would be possible to reconstruct the unknown word by an alternative approach, the text analysis phase would be unable to use it, because no further information about the word is available.

In the *DAD* we have analyzed several different distance measures for string comparison [Weigel 94b]. As result we decided to use a generalization of the well known weighted edit distance. This distance has the following important properties. It reflects the differences between strings in the context of OCR very closely. Furthermore it resolves typical normalization problems of the usual weighted edit distance but also of probabilistic distance measures.

Using a trie for representing the words of the vocabulary, we have developed a fast search algorithm integrating several static and dynamic heuristics for search space reduction. Determining adequate weights for elementary edit operations is an open research problem [Bunke 92]. We have

18. see Section 2.5.2
19. A lexicon rarely contains all words that can occur in a document.

developed an iterative learning algorithm for the costs of the elementary edit operations based on a kind of gradient descendant approach. First experiments show that this method can be successfully employed to improve the overall recognition accuracy. In the following we describe our solution in detail.

2.5.1 The context of our lexical postprocessing system

We first recapitulate the context and motivation of our approach for lexical post processing of OCR results. The process of automatic recognition of printed or handwritten input can usually be divided into several phases [Suen 86]. Typical phases are preprocessing for the purpose of normalization and for rough information extraction, segmentation, feature computation and classification. Especially segmentation is a critical phase. Frequently it is not possible to find only one alternative for splitting up the text into words and the words into characters. Instead several segmentation hypotheses have to be taken into account by the subsequent stages [Peleg 79].

Further ambiguities arise from classification. During the classification phase the segments which have been identified as possible characters are recognized. Again, it is impossible to identify the correct character each time. In such a situation it is often better to return several character hypotheses with confidence values instead of only the best, possibly false, one.

Because of all these ambiguities as the overall result so called *character hypotheses lattices* for each input word are returned ([Peleg 79], [Weigel 94]). A character hypotheses lattice represents a possibly large set of candidate strings for the corresponding input word. In a lexical postprocessing phase it is necessary to identify the input word by comparing the candidate strings of the lattice with the words of the vocabulary.

Developing a strategy for lexical postprocessing, requires investigations about the demands of such a phase. Important demands are:

- Efficiency: The time requirements are very important. Lexical postprocessing is a time consuming stage of the recognition task, especially if a large vocabulary is used.
- Correctness: The underlying model for comparing the candidate strings with the words of the dictionary must be adequate for the task of automatic recognition of text input. It should be able to handle all important kind of errors which could occur. Furthermore important information available like confidence values for the character hypotheses and error probabilities about character confusions has to be integrated and used.
- Adaptability: The approach should be able to adapt its search strategy to the quality of the input, to the specific characteristics of the preceding recognition stages and if necessary to the characteristics of the input writing.

In the following we describe a strategy for postprocessing handwritten or machine printed input which takes the above mentioned demands into account and has proven to work well in our application domain. In Section 2.5.2 we describe the fundamental depth first search of the algorithm. It is expanded by integrating substitution, insertion and deletion operations in Section 2.5.3. In Section 2.5.4 the computation of the confidence values for the word hypotheses is discussed based on some kind of weighted edit distance. The overall matching algorithm is further speed up by the use of heuristics described in Section 2.5.5. To get a self-adaptable system we propose an automatic learning algorithm for the costs of the edit operations in Section 2.5.6. First results are shown in Section 2.5.7 and we conclude with a summary and an outlook of our planned activities in Section 2.5.8.

2.5.2 Fundamental search strategy

An important aspect of the lexical postprocessing approach is the representation of the vocabulary. An often used structure is a trie [Appel 88], which is a prefix oriented representation of the words.

This structure allows an efficient realization of our matching strategy.

As mentioned before, the result of the preceding stages of the recognizer is a *character hypotheses lattice* (*CHL*) for each input word. A *CHL* is an acyclic directed graph as shown in Figure 15. Each node corresponds to a part of the input word identified by the segmentation algorithm. It is labeled with character hypotheses and their corresponding confidence values. The character hypotheses results from the classification task of the corresponding part or segment of the input word. The different paths through the graph are due to different segmentations of the input word. This way a *CHL* represents a set of candidate strings which have to be compared with the words of a vocabulary *L*.

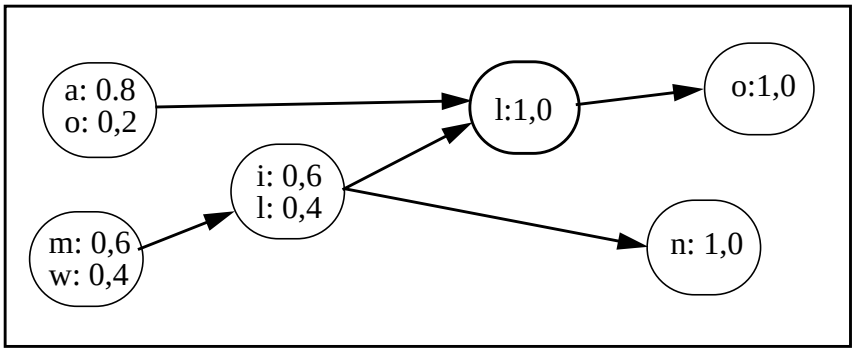

Figure 15: CHL

The fundamental idea for matching a *CHL* against a trie is based on a depth first traversal. Precisely, the paths of the *CHL* are traversed in a depth first strategy. Assume a node N of the *CHL* is visited and the string s is found in the trie on the way to this node. Then all continuations of s by character hypotheses x_i of N for which sx_i is a prefix of a word of the vocabulary are build up and stored in a string set S. For all strings t of S the search is recursively continued by visiting all successor nodes of N one after another with the string t. If a path in the *CHL* is completed and the string found this way is also a word of the vocabulary, indicated by an *end node* in the trie, a possible hypothesis for the input word is found. In the following this part of the overall matching strategy is called the *normal search.*

A special treatment is necessary for nodes in the *CHL*, which do not have any character hypotheses. Such nodes will be called *wildcard nodes*. If visiting a *wildcard node W* during the depth first search, all characters *y* found in the trie, which are possible continuations of *s*, are used to build up new string hypotheses as before. For each of these hypotheses the search in the *CHL* will be continued at the successor nodes as described above. In realistic applications wildcard nodes can occur frequently. Therefore the strategy of *wildcard expansion* can become very time consuming. To speed up the algorithm, a global variable defines the maximal number of wildcards which can be expanded on one search path through the *CHL*. That means if a wildcard node in the *CHL* is visited, wildcard expansion will only occur, if the number of wildcard expansions necessary to build up the string found on the way to the node is lower than a threshold *Maxwildcard*. Otherwise the search is not continued at this node. Carefully applied, this heuristic does not reduce the recognition rate significantly because usually only few wildcard substitutions occur on the right path.

2.5.3 Insertions, substitutions and deletions

Text as input[20] is a very unstable, highly variable and ambiguous signal. Therefore it is necessary to

20. Especially if written by some human being.

handle problems like insertions, deletions and substitutions of characters during a match of a *CHL* against a trie[21]. In this chapter a possible integration of these operations into the overall matching approach will be discussed.

Assume during the search a node N of the *CHL* is visited with the string s. For all possible continuations of s by character hypotheses of N, stored in the string set S, the search procedure will be called recursively for all successors of N as described above. This way the normal search with wildcard expansion is executed. When this search is finished for all strings of S some extra operations will be carried out at the node N:

- Substitution: For all strings *t* of S the first extra operation replaces the character hypotheses at each successor node of N by all those characters, which are not already member of the hypotheses set of the corresponding node, but an allowed continuation of the string *t* according to the transitions in the trie[22]. Subsequently the new character hypotheses are handled as usual hypotheses and the search is recursively continued by adding the new character hypotheses to *t.* Obviously successors of N which are wildcard nodes are not considered by this extra operation.
- Insertion: For all strings *t* of S the next extra operation inserts additional characters. Therefore a new successor of N, an extra node *E*, is build up, which contains all characters which are possible continuations of the string *t* in the trie. The successors of node *E* are the same as the successors of N. Subsequently the search continues at the node *E* and recursively proceed at the successors of N (and *E*). *E* is deleted when the search returns to N.
- Deletion: For all strings *t* of S the last extra operation skips every successor node *U* of N in the *CHL* and continues the search recursively at each successor of *U* in the usual way.

The algorithm described so far computes some kind of edit transformation between the strings represented by the *CHL* and the words of the lexicon. It is not based on a dynamic programming approach [Wagner 74], but on a recursive depth first traversal. A matching algorithm integrating the extra operations as described above has some problems with the combinatorially exploding size of the search space. In one of the following sections we propose several heuristics which speed up the search without affecting the recognition rate significantly. This way it outperforms our approaches based on dynamic programming.

2.5.4 Weighting the word or string hypotheses

So far, only the strategy to direct the search for possible word or string hypotheses is described. It is further necessary to assign measures of belief to the strings or words found this way, to be able to select the best one. We base the computation of these measures of belief on the confidence values of the characters in the *CHL* and the edit operations used for the match. Assume again that a string s is found on the search to node N in the *CHL*. Now the question is, which measure of belief should be given to s? For this purpose, we use some kind of weighted edit distance, which is generalized by allowing negative costs for edit operations as well as integrating the confidence values of the character hypotheses at the nodes of the *CHL* ([Weigel 94],[Weigel 94b]).

First we assume that s is found by a normal search without any extra operation including wildcard expansions. To compute the measure of belief for s we multiply the confidence values of the character hypotheses x_i used for building up s with some edit costs defined by $\gamma(x_i \rightarrow x_i)$. The products are subsequently added resulting in the measure of belief for s. According to our previous

21. At the moment we also try to incorporate merges and split ups.
22. Afterwards, the original character hypotheses are restored.

work we usually assign negative costs to operations of the form $x \rightarrow x$.

If s is found by a search with some wildcard expansions, the computation of the belief measure is done in a similar way except for the wildcards expansions. For each expansion by a character x_i, a value $\gamma(\omega \rightarrow x_i)$ is added[23].

In a similar way, specific values for the extra operations are determined, which are subsequently added to get the overall measures of belief of the word hypotheses. For substitutions of the character hypotheses of a node *N* by a character x_i, the confidence value of the best hypotheses *y* of *N* is multiplied with cost $\gamma(y \rightarrow x_i)$ [24]. If the deletion of a node *N* occurs, the confidence value of the best character hypothesis *y* of *N* is multiplied by $\gamma(y \rightarrow \varepsilon)$ and the result is subsequently added[25]. If *N* does not contain any character hypothesis only $\gamma(\omega \rightarrow \varepsilon)$ is added. If an insertion of a character x_i occurs during the match with s the value $\gamma(\varepsilon \rightarrow x_i)$ is used for this operation.

The measure of belief for a word hypothesis is a kind of edit distance, with some extensions. Negative values for the costs of edit operations are allowed and the confidence values of the involved character hypotheses of the *CHL* are integrated [Weigel 94].

2.5.5 Speeding up the search by the use of heuristics

In its fundamental form, the proposed strategy matches a *CHL* or the candidate strings represented by a *CHL* against the words of a trie based lexicon. This is done by finding the maximal confidence measure for some candidate string by applying the edit operations, proposed in the sections above. One main problem is the complexity of the search space (Note, we do not use any dynamic programming approach to speed up the task). This will be resolved by the use of heuristics, which speed up the algorithm without reducing the recognition rate significantly.

The most time consuming operations are extra operations (substitutions, insertions, deletions) because they expand the search space by the order of magnitudes. Therefore, the first heuristic is a condition, which is checked before the extra operations are started at a node. Extra operations will be only executed if this condition is fulfilled. Assume a node *N* is visited and the string *s* is found on the way to it. Then the condition looks as follows:

$$\frac{LoConst + LoCost}{Length} > LoThresh$$

LoCost is the measure of belief for the string *s* found on the path to *N*. *Length* is the length of *s* and *LoTresh* is a predefined threshold. *LoConst* is also a user defined parameter which is necessary, because otherwise an extra operation at the beginning of a path in the *CHL* is prevented.

With the next heuristic, extra operations are prevented, if a good word hypothesis is already found by completing the actual search path. The condition is:

$$MaxMeasure < HighThresh$$

MaxMeasure is the confidence value for the best word hypothesis found by completing the actual path in the *CHL* before the extra operations at the actual node *N* are executed, divided by the length of this word (for normalization issues). Because of its recursive structure the program can execute extra operations at successors of *N*. *HighTresh* is a user defined constant, which should be high enough to prevent that important edit operations will not take place.

Also the execution of the normal search at node *N* (before applying extra operations) should be

---

23. ω expresses a wildcard node.
24. A direct extension is possible by integrating all character hypotheses of the actual node to determine the value of γ instead of only considering the best one.
25. ε expresses the empty string.

limited by a condition as follows:

$$\frac{LoConstNormal + LoCost}{Length} > LoThreshNormal$$

LoConstNormal and *LoThreshNormal* are constants similar to *LoConst* and *LoThresh*. But this condition prevents the execution of a normal search step. By this condition succeeding extra operations at the actual node are also excluded.

So far all conditions given are static. Therefore these conditions cannot adapt themselves to the quality of the recognition results and it is necessary to make these conditions not to restrictive, because otherwise no word will be returned if the recognition results are not so good. Therefore further dynamic conditions are used which become more and more restrictive according to the measure of belief of the best word hypothesis found so far during the search. There are two dynamic conditions used by our algorithm:

$$\frac{LoConstDyn + LoCost}{Length} > GlobalMaxMeasure \times Par$$

$$\frac{LoConstNormDyn + LoCost}{Length} > GlobalMaxMeasure \times ParNorm$$

The first condition must be fulfilled if the extra operations should be executed and the second condition controls the execution of the normal search steps. *GlobalMaxMeasure* is the dynamic part of these unequations. It contains the confidence value of the best word hypothesis found by the search algorithm up to this moment, normalized by the length of the corresponding word hypotheses. The better this hypothesis is, the more restrictive are the conditions. Therefore, if a good word hypothesis is found only few further paths in the *CHL* will be investigated. *Par, ParNorm, LoConstDyn* and *LoConstNormDyn* are predefined constants.

2.5.6 Learning the values of γ

The values of the function γ assign the costs to the different edit operations, which are applied to find a word hypothesis. We define that high positive values reflect a low probability and negative values a high probability of the corresponding edit operations [Weigel 94b]. To find adequate values for γ is of high importance. For a user it is nearly impossible to find such values. Using an alphabet of about 80 characters more than 6000 values have to be determined and carefully harmonized with each other.

The problem of automatic determination of adequate edit costs is still an important research area [Bunke 92]. In [Bunke 93] an interesting basis for an approach is proposed, but its time requirements are enormous, so that it is not applicable in our domain. Other usable solutions are not known to the authors. Therefore we developed an automatic iterative supervised learning scheme, which automatically computes the values of the function γ. This algorithm will be presented in this section.

Assume a training set of input words $W = \{w_1, w_2 \ldots w_n\}$ is given as well as the corresponding set of *CHL*s $C = \{c_1, c_2 \ldots c_n\}$. Starting with some initial values for γ the *CHL*s of the words in *W* are given to the matching algorithm described before. If the correct word is recognized for an input *CHL*, nothing happens. Otherwise, if a wrong word hypothesis is returned, the function γ is adapted. This works as follows. Assume that *CHL* c_i is given to the matching algorithm and the word hypotheses h_i is returned, which is different to the correct word w_i. Further assume that the measures of belief for h_i is computed by

$$\sum_{x_j \to y_j} A_j \times \gamma(x_j \to y_j)$$

as described above. A_j describes the factor by which the costs of the corresponding edit operation are multiplied and subsequently added to the measure of belief of h_i. If the edit operation is not used for the match A_j is equal to zero. We assume that each A_j describes the overall factor of the corresponding edit operation. That means if an edit operation $\gamma(x_j \to y_j)$ is used several times in the match, the corresponding expressions are all combined into the single term $A_j \times \gamma(x_j \to y_j)$ [26].

To get information about what has to be learned, we match c_i with a lexicon containing only the correct input word w_i. To make sure that in most cases a solution in form of a sequence of edit operations transforming the *CHL* to w_i is found, the parameters for search space reduction are adjusted more liberate than before[27]. For the match found the confidence value for w_i can be computed by $\sum_{x_j \to y_j} B_j \times \gamma(x_j \to y_j)$ [28].

Given $\sum_{x_j \to y_j} A_j \times \gamma(x_j \to y_j)$ and $\sum_{x_j \to y_j} B_j \times \gamma(x_j \to y_j)$

it is possible to adapt the costs for the edit operations, because the first term describes the costs of a false edit sequence whereby the second the costs of a correct one. You can argue that the costs for edit operations which contribute more to the wrong hypothesis should be raised whereby the costs for edit operation which contribute more to the right hypothesis should be lowered. An adaptation of the edit costs take place according to the formula:

$$\gamma(x_k \to y_k) = \gamma(x_k \to y_k) + (A_k - B_k) \times \varepsilon$$

ε is a positive constant which can be reduced if the training proceed to prevent typical triggering problems. By this formula, the costs for edit operations which contribute more to the right hypothesis than to wrong are reduced. On the other hand costs of edit operations contributing more to the wrong hypothesis are increased.

Some special cases have to be considered separately. First if no word h_i is returned, we assume that all A_j are equal to zero. Also if the reference search in the trie containing only the word w_i fails, all B_j are assumed to be zero. In these cases the updating function for the edit costs can be used without changes.

2.5.7 Experiments

In our experiments we used a document database of about 90 scanned business letters many of them of bad quality. For all documents the recognition results, i. e. the *CHL*s, are stored. Together the documents contain about 9500 words. The document set is divided into a training set containing about 8300 words and a test set containing about 1200 words. The training starts with some initial values for the edit costs, carefully selected during experiments, but with the restriction, that edit operations of the same type (deletion, substitution etc.) do have the same costs.

Training of the edit costs was done by running through the train set for 20 times. After each traversal the intermediate values for the edit costs are tested against the test set. In Figure 16 the number of words correctly recognized in the test set after each training step are shown. Starting with 750 words before training, after 20 traversals through the training set about 915 words are correctly recognized. In Figure 17 the number of wrong words recognized after each training step is shown,

---

26. This is easily done by $A_j \times \gamma(x_j \to y_j) = C_1 \times \gamma(x_j \to y_j) + C_2 \times \gamma(x_j \to y_j) \ldots$ with $A_j = C_1 + C_2 \ldots$.

27. This is no problem because the lexicon contains only one word and we do not have any time problems.

28. Again we assume that each B_j describes the overall factor of the corresponding edit operation.

starting with about 235 and decreasing to 80. In Figure 18 the number of rejections is presented. Here the number is relative stable over the training period, between 200 and 255.

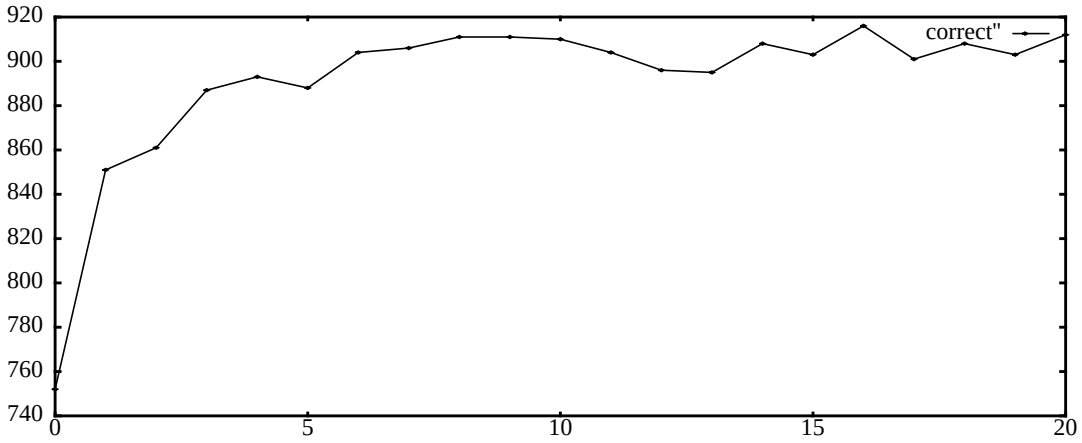

Figure 16: Correct words

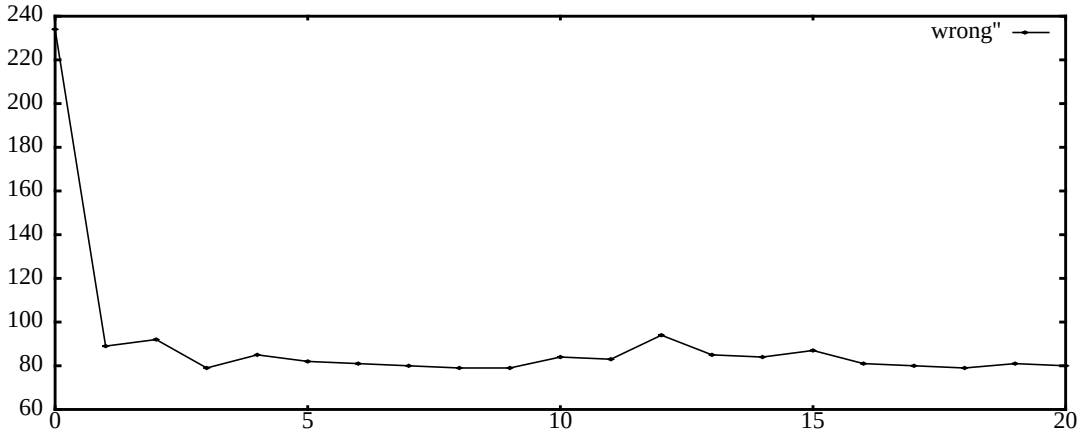

Figure 17: Wrong words

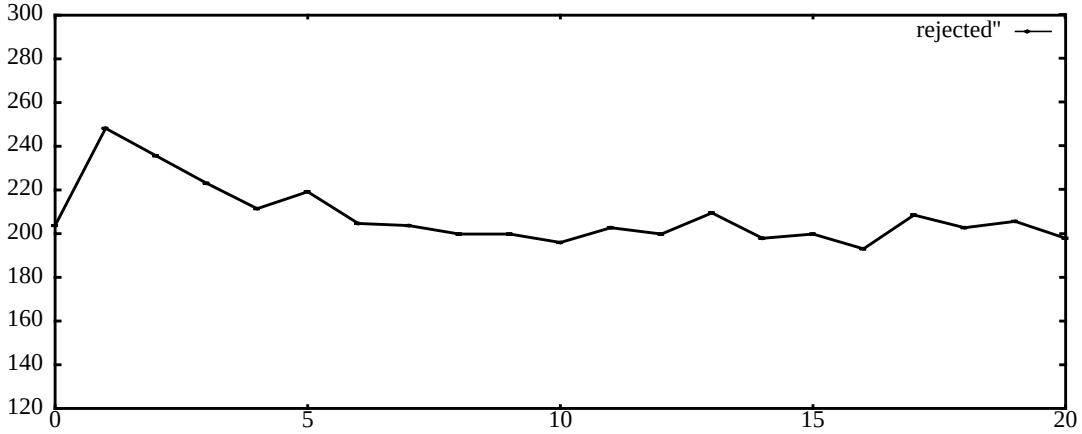

Figure 18: Rejections

The improvements of the recognition results due to the training procedure are obvious. Only about 6 training steps are necessary to achieve the best results. The experiments we conducted on a Sun

Sparc Station 20 with a C++ implementation of the algorithm. On this machine about 9 words per second can be processed.

We think further progress is possible by building font specific postprocessing modules, which models the specific errors of different document types more adequate. Also the integration of additional edit operations for merges and split ups will improve the recognition accuracy as well as the efficiency of the learning algorithm.

2.5.8 Summary and future work

We proposed an approach for efficiently postprocessing printed and handwritten input by the use of lexical knowledge. It is based on a complex strategy to compare the candidate strings in a character hypotheses lattice with the words of a vocabulary. By the use of several heuristics it is possible to reduce the time complexity by the order of magnitudes without decreasing the recognition accuracy significantly. To allow an automatic adaptation of the algorithm to the strengths and shortcomings of other recognition phases an iterative automatic learning algorithm for the costs of the edit operations was outlined. Experiments reveal that a significant reduction of the error rate can be achieved by this method.

At the moment we are working on methods for automatically adapting the search parameters to get a complete self-adapting system. Furthermore we try to handle segmentation errors like character split ups and merges by integrating additional edit operations for these cases in a straightforward fashion.

Also some theoretical investigation of the properties of the learning algorithm for the edit costs are under way. First results are presented in Appendix 1.

An open question strongly related to the time behavior of a string correction module is the size of the vocabulary used. It has to be clarified if it is of benefit to include all possible words or to use only the most important ones. This depends strongly on the demands of the text analysis phase but also on the quality of the input data. Using a large dictionary poses efficiency problems and can result in unwanted word substitution faults. On the other hand, a small vocabulary does not contain all words in the text and produces many errors or at least no results, if unknown words are feed into the system.

To handle words not appearing in the lexicon, postprocessing approaches based on incomplete knowledge about the vocabulary (e.g. n-grams) should be considered [Hull 88]. For us this approach does not seem to be very ingenious because unknown words can not been handled by the succeeding text analysis phase.

Motivated by the use of concurrent lexical postprocessing algorithms, a further task of the text recognition process could combine different word recognition results to produce a uniform output for the text analysis steps. Again, the homogenization is only one aspect, the other goal is an improvement in overall word recognition accuracy. Most statements made for voting on character recognition level also hold for voting on word recognition level, especially those concerning the three representation levels and the induced combination methods. Again, it is possible to compare different approaches ranging from simple majority vote to more sophisticated methods involving probabilistic reasoning.

2.6 Reactive plan generation

The sections above have listed the typical analysis tasks of a document analysis system together with some of the approaches used to solve the corresponding problems. So far, nothing was said about the combination of these approaches. This is a task itself and will be described in this section.

2.6.1 The task of planning the analysis flow

There are a lot of problems in a document analysis system which make its configuration a difficult task.

- Most subtasks of text recognition and analysis could not be handled with a single general solution algorithm. On the contrary, it is crucial to built algorithms for input data with different characteristics, so there will often be a set of specialized approaches; e.g. one OCR is better suited for font types with serifs whereas another is more reliable on sans serif fonts.
- Document analysis in our domains is a very complex problem, where a wide variety of input documents with different characteristics have to be handled. Because of this complexity, it is not possible to state beforehand which algorithms can be best applied for the different subtasks. Not even the order in which the subtasks have to be processed may be obvious in advance.
- Most analysis algorithms return alternative results. Therefore, a combinatorial explosion of search space must be prevented.
- Most results of the analysis algorithms are vague and therefore attached with different measures of belief. Furthermore, often there are no or even false results.

All these problems motivate the development of a reactive plan generation module and its integration into a document analysis system. The goal is to build this reactive plan generation module being able to combine the specific recognition algorithms in a way, that optimal results are achieved by the controlled system, with a reasonable use of system resources.

2.6.2 State of the Art

Because of the problems pointed out above, a static analysis sequence does not work in practice. For many tasks there are various analysis algorithms with different applicability, depending on document characteristics. The idea to apply all available analysis algorithms in all possible combinations is unfeasible, because of the time constraints imposed on the overall analysis. Only for some specific subtasks, it seems to be useful to apply all the corresponding specialists. In such cases, a voting component could be used, to combine the different results and thereby preventing a search space explosion. This approach is chosen for the character classification task in our system (see Section 2.4).

So, what we need is a component which dynamically plans the document analysis process. It should consider the results of each algorithm and estimate if progressing in the analysis flow seems possible or if the current results need improvement or refinement beforehand. An analysis system working like this will always try to guarantee best possible result quality with respect to a global goal. Furthermore, it constrains the number of result alternatives obtained by successive knowledge sources.

There have been some rule-based approaches to organize limited subtasks of document analysis, e.g. segmentation [Fisher 90] and labeling [Niyogi 86],[Kreich 91]. Our plan generation should treat complex interactions between diverse stages of the analysis. Therefore a comparison with the following systems seems more adequate:

ABLS (Address Block Location System, [Wang 86]) is a framework used to determine destination address blocks on arbitrary mail peaces. ABLS contains a dependency graph to describe the interdependencies of all tools in the framework, separate knowledge rules with selection and utilization criteria for each tool frame, and a blackboard for the exchange of tool results. Control rules are used to relate knowledge rules and analysis states and strategy rules direct the search of result hypotheses. The inference engine uses a generate-and-test procedure, pursuing the occasionally

best hypothesis.

Another system following the generate-and-test strategy is GRAPHEIN [Chenevoy 91]. It is also blackboard-based and able to follow alternative strategies for one task. The highest level of control is a “selector” that decides which specialists are applicable in a specific situation. This meta-knowledge consists of control rules containing methodologies of how to solve subgoals of a problem.

Basic work in this field was also done in the WIDAN project [Fein 92],[Kasper 94] where a model-based control was proposed. The analysis system consisted of two kinds of problem solvers: analysis problem solvers which perform basic tasks and control problem solvers which organize subtasks by coordinating analysis problem solvers or other control problem solvers. The problem solvers itself are arranged in a taxonomy. The logical coherency between them are described in analysis models which use operators to define alternative strategies. The goal was to avoid errors rather than to react on errors. However, it is difficult if not impossible to describe problem solver behavior in a precision useful for this purpose.

InfoPortLab [Bayer 94] is a document analysis system where a semantic net language FRESCO is used to describe properties and relations of certain algorithms for layout extraction and text recognition. These descriptions are collected in a knowledge base. The inference engine uses an “aggregated method” from the model which describes the dependencies between “basic methods,” i.e. analysis algorithms. According to the results on a blackboard, applicable algorithms from the model are gathered, their respective usefulness rated, and the most promising one executed. The control component of InfoPortLab is based on technology developed in the WIDAN cooperation project between DaimlerBenz and DFKI.

The above systems all use a blackboard for exchanging knowledge and results between plan generation component and analysis algorithms. Dynamically planning and controlling the analysis is done by establishing a dynamic interlocking of analysis approaches. Since the blackboard mechanism is very useful for this purpose we also use it in our approach which is described in the next section. This description is centered around the blackboard data structure as the basic communication architecture.

2.6.3 Planning and control scheme

In order to build a system with flexible problem solving behavior we need a planning component. It should be capable of recognizing approaches adequate in a specific analysis situation of a given document and treating the entanglement of the respective approaches. Only those approaches should be applied which seem necessary or useful in certain situations with respect to the global analysis goal. By monitoring progress in the analysis and reacting on problems within certain analysis approaches an improvement of analysis can be achieved.

An obvious idea for solving this task is to build a collection of typical problem cases together with solutions. During analysis one then can check each analysis situation against known problem cases and eventually “repair” the analysis flow. However, this assumes relatively static problem descriptions which are not sufficient for our purposes, since a problem might have different causes. But often the degree of a problem (e.g. percentage and number of misrecognized regions) gives hints about reasons and possible tuning methods.

Thus we need flexible and declarative problem descriptions together with the ability to compare these with typical situations during analysis. Since these situations can be defined by analysis history so far and both available and expected result characteristics, the control task involves tracking the analysis process and all intermediate results. If we can find suitable situation descriptions, the combination of approaches could be done at runtime by checking actual analysis situation and comparing against expectations. We could then infer how to continue the analysis.

This gives us both options for

- feedback loops between approaches — some approaches can be simplified if they have “knowledge” provided by a task normally solved later: e.g. the lexical postprocessing can detect problems resulting from segmentation and therefore give hints for redoing segmentation
- special postprocessing steps — sometimes a specific document might pose a problem solvable with support of an additional special approach: e.g. OCR problems sometimes can be cured by identifying the document font prior to applying a single-font classifier.

It would be expensive to propagate all kinds of data to every specialist since usually only few of them can actually use it or are interested in it. A better approach is to just inform knowledge sources actually affected by certain information.

This leads to a blackboard style system architecture (see Section 2 and Figure 19) consisting of three parts: knowledge sources, a common data structure for sharing results and information, and a scheduler which globally plans and directs the analysis efforts of all knowledge sources [Hayes-Roth 85], [Engelmore 88].

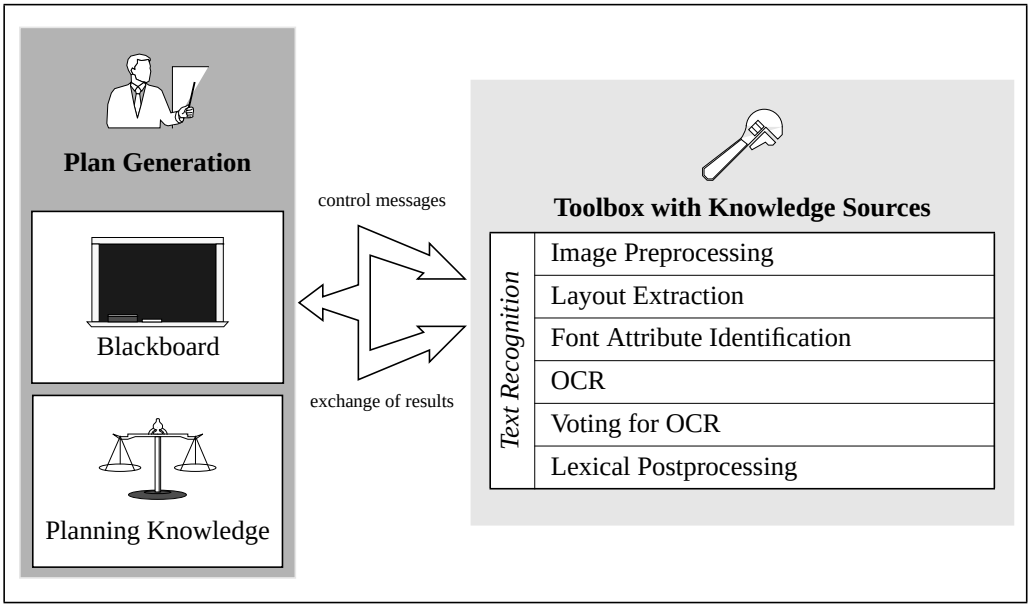

Figure 19: Relation between plan generation and other tasks

To enable a global control, each algorithm in the system provides a uniform interface with self-describing information, especially its preconditions for running and into what data it is interested. Besides this, a measurement of average reliability is useful when there are several alternative knowledge sources for a task, e.g. different character classifiers. The knowledge sources must also understand a special protocol for the communication with the controlling scheduler.

Each knowledge source collects information from a global data structure — the blackboard — and returns new information to it. At each time of the analysis process all available information is accessible from the blackboard together with the respective creators. For the global evaluation of data it is especially helpful if all data have some kind of credibility measure attached.

The scheduler has strategy knowledge for performing the analysis. It monitors the behavior of each knowledge source for a specific document and revises its analysis strategy after certain plan steps. By comparing credibility measures contained in the results of knowledge sources and applying internal control knowledge it makes assumptions about the overall analysis quality at a time. Depending on this quality together with a preset analysis scenario it then decides upon proceeding in the analysis process: acceptable quality will result in applying a knowledge source leading straight forward, whereas problems will possibly result in activation of specialized knowledge

sources depending on the derived cause.

2.6.4 Summary and future work

We have motivated a reactive plan generation component as the basic control part of a document analysis system. The main reasons are the absence of an unambiguous and static analysis sequence and the complexity of the results returned by all approaches.

The proposed component pursues a plan which is permanently refined by reacting on results returned by the approaches. This way, only the most promising approaches with respect to an analysis situation will be applied. On the one hand the sequential order of processing subtasks has to be identified. On the other hand for each subtask the decision between competitive approaches must be considered.

A first prototype for this kind of analysis control has been realized. Future work will concentrate on the development of elaborated strategies for the analysis. This requires investigating sample plans, determining their weaknesses, and working out improvements for the whole system. The latter can be achieved by an enhanced combination of currently available algorithms and by establishing the need for additional specialists and their proper integration.

3 System architecture

As mentioned in the precedent section, for each analysis (sub)task, several problem solving techniques are planed or realized, which are competitive or complete one another. They have different abilities and thus are qualified for specific analysis situations. In which situation which problem solver have to be activated is task of the plan generation module. The resulting system architecture consists of four main components:

- set of analysis specialists
- plan generation module
- knowledge base manager
- document archive manager

The set of problem solving algorithms are further divided into text recognition algorithms being discussed in this report (cf. Section 2), and text analysis algorithms being presented in a separate report [Baumann 95]. The plan generation module is introduced in Section 2.6 and its necessity for a successful analysis was pointed out.

The two system parts, namely the knowledge base manager and the document archive manager are not introduced in this report. The first one handles and interprets the knowledge about the analysis process while the second one is responsible for managing the analysis results of all problem solvers as well the final system output. The interesting reader is referred to [Baumann 95] and [Bläsius 95]. Figure 20 illustrates graphically the system architecture.

In general we can state that our system architecture is adequate for a large set of document analysis applications. This ability is rendered by the *modularity* of the entire system. Single analysis components can be easily exchanged or improved without modifying other components. A second feature is the separation of domain-specific knowledge and storage in a knowledge base. This separation allows an easy adaptation of analysis algorithms to different tasks or domains. Also, specific analysis strategies of the reactive plan generation module are stored in a declarative manner, allowing a problem-oriented analysis and therefore serving as a basis for a *document analysis shell.*

Currently, our text recognition system consists of one layout extraction routine, one font attribute identification routine, two OCR routines (commercial devices), one voting routine and one lexical postprocessing routine. Most of the routines can be activated with specific parameters which influence their analysis characteristics. For the plan generation module, a first prototype is available which uses declaratively stored plans to control the analysis flow.

The following example depicts a possible analysis scenario for the text recognition phase. The reactive plan generation component will typically start with a standard plan suitable for most kinds of documents in a domain. This will usually be something like image capturing, segmentation, OCR, voting and lexical postprocessing. After each plan step the global change on the blackboard is measured, e.g. by collecting the credibility measures of new result hypotheses. All intermediate credibility measures are combined and evaluated according to the control knowledge. The control knowledge will probably express, that an abrupt decrease of credibility between two knowledge source activations indicates problems in one of the former phases. E.g. OCR errors usually result from split ups or merges not detected during segmentation. The segmentation probably will have marked uncertainties in a way, so that inferring the cause for the problems is possible. The control knowledge describes solutions for each kind of analysis problem. An alternative plan to handle the OCR problems described above could be a resegmentation of split or merged characters followed by a restart of the OCR, voting, and lexical postprocessing modules.

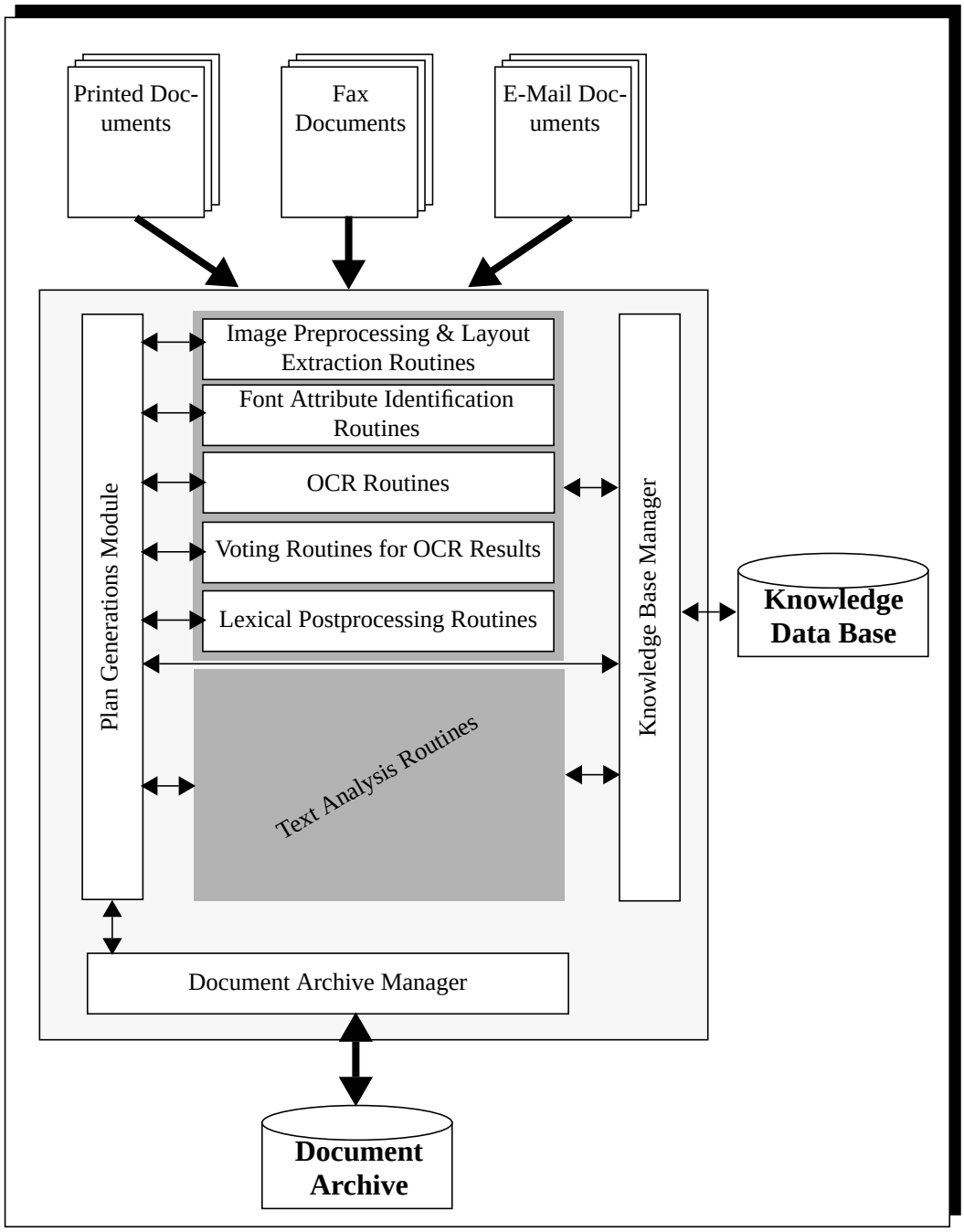

Figure 20: System architecture of our document analysis shell.

4 Testing & performance measurement (benchmarking)

Testing and performance measurement is a vital task for almost each project but also for many research areas. Beside others, it enables the management, the client, the scientist, or the developer to

- judge project progress,
- analyze system performance,
- discover problems or faults,
- compare alternative approaches.

A document analysis system is build up of many different modules. Testing and performance measurement is indispensable for the project success because it enables the developers to measure their progress as well as the quality of their system compared to competing ones. It furthermore builds a basis for a systematic identification of problems, which is perhaps the most important source of improvement. Obviously in a research project we are not so interested in testing maintainability and other typical questions which are important from a software engineering point of view[29]. Instead we are mainly interested in the capacity of our solutions, e. g. the number of correctly recognized characters or the number of missegmented words.

At the current state of research, testing and performance measurement in the area of document analysis is only rudimentary considered. Despite many researchers describe some experiments, most of them for single modules resolving only subtasks of the overall document analysis problem, the test methods are rarely useful to serve as a basis for a more general approach. In this context the work of *ISRI* is worth mentioning ([UNLV 93], [UNLV 94]). The researchers at this institute are working on fundamental techniques for testing systems in the area of document analysis, especially for OCR systems. The lack of useful methods for benchmarking is the reason why we have to develop methods, which gives us all the information mentioned above.

In the first step to develop a strategy for testing we have to specify what is the intended in- and output of the system under consideration. Then adequate metrics have to be defined which allow a judgement of the different properties of the system. In spite of a complex system, e. g. a document analysis or text recognition system, we are mainly interested in the overall system performance. With respect to this goal, the performance of single modules can never be analyzed only in isolation, because the combination of several good modules must not necessarily result in a better system than the combination of modules which are not so good. This is especially true, if the first ones are not harmonized to each other. Nevertheless, also tests have to be developed for system components (sets of modules). This enables a researcher to analyze a single component and helps him in his decision how to proceed in his work. Furthermore such tests help other developers to decide which solution they should integrate into their system.

Another important task is to build up test sets which reflect the real input data as good as possible. Because usually a test set can not comprise all possible input alternatives (possibly with the right distribution), it is a challenging task to build up such sets.

In the following sections we want to discuss the tasks, challenges and possible approaches for testing layout extraction, OCR, voting on character level, lexical postprocessing and plan generation. In our view testing plan generation covers testing of the overall system performance, which means testing the text recognition system in this research report. Testing of the whole document analysis system is considered in [Baumann 95], where also the overall goals of the system are specified.

---

29. This does not mean, that we are not interested to fulfill these properties.

4.1 Testing layout extraction

Documents digitized by an optical scanner or received via fax are represented as binary images. To extract information from the image, it is necessary to group image points (pixels) to segments. In this sense the layout extraction is the first analysis phase generating a new representation of the binary image, that makes various kinds of knowledge explicit.

The main goals of the layout extraction are the generation of block, line, word, and character segments and the separation of text from nontext segments. Character segments are input of character classifiers recognizing the character shape as a symbol of a given alphabet. Word segments are a basis of the word verification components identifying a character sequence as a legal string or known word. The line and block segments are necessary to calculate several features for text recognition (e.g. half- and baseline of a text line), to determine the context of characters (e.g. font type), and to identify the reading order. For text recognition only text segments are relevant. Separation of text from nontext avoids unnecessary and time intensive recognition work. At the same time, text analysis will not be confused by incomplete and erroneously recognized "graphic strings".

Because all subsequent analysis phases depend on the results of the layout extraction, it is important to perform a correct and reliable segmentation. A successful layout extraction is a necessary precondition for the other steps in the document recognition process.

To measure the performance and estimate the quality of a layout extraction algorithm we distinguish the following evaluation criteria:

- Completeness of text segments: The layout extraction algorithm must detect all text regions in the document (binary image). Undetected regions are for ever lost for the subsequent analysis phases.
- Soundness of text segments: Every region identified as a text segment must be really a text region. Graphic regions identified as text segments complicate the task and aggravate the results of text recognition.
- Exactness of text segments: The position and amount of pixels belonging to one extracted text segment must be exact i.e. only relevant pixels should be grouped. Overlapping with pixels of adjacent text segments or of the background is not allowed.

Results of layout extraction consist of pixel classification[30] and page segmentation[31]. Estimating the quality of pixel classification is moderately difficult: the solution is unique, and one can simply compare the classification result with pre-stored "ground truth" data, including for each pixel in the binary image its corresponding correct class.

Unlike for pixel classification, evaluating page segmentation is not straightforward. The optimal segmentation is not necessarily well defined and unique for a page [Randriamasy 94]. Comparing the results of an image segmentation algorithm with the expected results can not be realized using an exact matching. Instead of this "pure" identity matching a sophisticated similarity measure for two (or more) objects is necessary.

A method capable of providing an explicit error specification can help us to improve our currently developed segmentation algorithm and to measure its performance and its quality compared to competing ones. Moreover, a segmentation algorithm must be tested on hundreds of input pages, which stresses the need for an automatic tool [Randriamasy 94].

An automatic image-level, segment based evaluation of a layout extraction algorithm compares results with predefined ground truth information. The ground truth zones and the created real zones should be represented in the same way. Possible representations of zones are rectangles (bounding

30. in text or nontext pixels.
31. i.e. segmentation of the binary image in text (characters, words, lines, blocks) and graphic objects.

boxes) or polygons. The similarity measure depends directly on the chosen zone representation.

Manually labeling of real documents is a very hard and time consuming task. Thus testing (benchmarking) the layout extraction should begin first with providing a tool for automatic generation of ground truth data from a binary image. Such synthetic data consists of a description of possible correct segmentations. It should involve block, line, word and character ground truth information. Important ground truth information for the layout extraction are: position, bounding polygon (or contour description) and type (text, nontext). The segmentation results are then matched against the ground truth information. The approach here is to compute the overlapping and the similarity of every “ground truth” zone with all “real” zones and vice versa.

The output of the testing (benchmarking) tool should provide a multi-level output for a given document test suite and its corresponding ground truth information [Randriamasy 94]:

- Overall output of the test suite: a numerical statistic should be provided over the test suite. It consists essentially of the arithmetic means of the results of the page-level output.
- Page-level output is provided both for the “real” document and for its corresponding ground truth data. It includes error frequencies such as number of undetected regions, number of merged or split regions etc.
- Region-level output: is only detailed for the incorrectly segmented ground truth zones. For each of them, the testing (benchmarking) tool provides: type, position (location), number of pixels missed by segmentation, number of regions it has been split in, the diagnosis on the splittings and merges done on that region.

4.2 Testing OCR and voting on character level

Most publications about OCR and voting of OCR results include some tests, which measure the

- character recognition rate,
- character misrecognition rate,
- character rejection rate,
- time consumption

of their solutions. Though these are the most important quantities, it is obvious that extensions to these fundamental indicators can be defined to describe how well a system works. For example different faults can be weighted by different costs to value their importance. Further result classes like split ups and merges of characters can be considered.[32] As result we can say that there is a wide variety of possible metrics to describe system behavior quantitatively. Which is the best one is impossible to decide in general and depends strongly on what we are mainly interested in.

Though there are many publications on performance measurements of OCR and voting systems, the results are difficult to compare. This is mainly caused by the fact that no uniform test approach is available. Perhaps the main reason for this are the different test sets used for the experiments.

Building up realistic test sets for performance measurement is a time consuming and therefore expensive task. Large sets of ground truth data are necessary. At the current state of research no perfect format for the ground truth data is defined. For example, is storing the ASCII text enough, or is it necessary to add zone information for the blocks, words, or even characters? Furthermore, should

32. Here we are confronted with faults which could be imposed by the segmentation phase. If we are interested in the behavior of the complete OCR system (including preprocessing, segmentation...) such faults are important test quantities. If only the classification phase is analyzed, an approach which is able to recognize merged characters is handicapped. This example shows how much the test strategy can be influenced by the relationships between the different tasks of the overall problem, and that a decision is necessary if a module should be tested in combination with its preceding phases or not.

typical document information like the predominating font, the print quality etc. be also stored? At the moment only one set of test data is publicly available, the Haralick-CD [UW 93].

Directly related to the problem, which information should be stored, is the wide spread of test algorithms which can be applied to measure the performance. For example some researchers use the presegmented words or characters thereby reducing the error rate because no segmentation errors can occur. Using presegmented characters as input data, the test algorithms become very simple. They must only count the correct and incorrect recognized characters as well as all rejections. Solutions for test data which are not presegmented are much more difficult. Elaborate matching algorithms must be developed and applied, comparing the output of the recognition system with the ground truth data. Handling character and word segmentation errors adequately is only one of the problems encountered here.

It becomes more and more obvious, that test sets and test procedures have to be standardized to support further progress in this research field. An interesting step in this direction are the efforts of the *ISRI* ([UNLV 93], [UNLV 94]), which conducted annual tests of the accuracy of OCR and voting systems. These test are only for general purpose systems. It seems to be necessary to define also tests for specific applications.

At *DAD* we are also developing test algorithms for OCR and voting results. We need algorithms for our special result structures including particularities like segmentation alternatives and character hypotheses sets. The test algorithms should not only measure our own progress, but also help to identify predominant errors made by the system. Furthermore, we build up our own set of test data, some of it synthetically generated by the use of distortion models, others based on public available test sets. Building up our own test set makes us independent from the rare public available sources of test data, which are seldom adequate for our research. Furthermore, we are forced to build up test data of documents from our application domain, which has its own characteristics on which the system has to be trained and tested.

4.3 Testing the lexical postprocessing

The objective of performance measurement of the lexical postprocessing and a word voting phase is to determine the

- number of correct recognized words
- number of misrecognized words
- number of rejections
- time consumption

or extensions of these fundamental quantities. Most of our discussion in Section 4.2 is also true for testing solutions for lexical postprocessing and is not repeated here once again. It should only be noted, that again the ISRI conducts an annual test where the word recognition capacity of different systems[33] is analyzed [Rice 94].

4.4 Testing the Plan Generation

As described before the plan generation module should control the overall document analysis system, but it must also be able to direct only parts of it, like the text recognition phase. Testing the plan generation module can be split up into two subtasks:

- plan testing
- system result testing

33. In this case OCR systems.

Plan testing consists in measuring characteristics of the solution plan, e.g. time needed for problem solution, number of analysis steps (invoked knowledge sources), storage requirements of the controlled system etc. These measures do not describe the result quality of the controlled system, but rather are measures for how the solution was achieved. A challenging problem for plan testing is to define a “ground truth” plan, i.e. the optimal plan for a given analysis problem, with which the actual solution could be compared. Also, for this comparison adequate metrics have to be defined.

It is obvious that the results returned by the controlled system also determine the quality of the plan generation module. Only a good plan generation strategy is able to employ the right knowledge sources in an efficient manner to resolve the problem at hand. Testing of the system results can be done by using the performance measurements defined for the corresponding back-end specialist. Thus, progress in the plan generation module is achieved if the results of the controlled system have improved by only changing the planning strategy.

To value a plan generation module both the measurements of plan testing and of system result testing have to be considered. Generally which measures are more important depends on the application and the goals of the project and has to be investigated with care.

5 Conclusion

In this report the current text recognition activities at the *DFKI's* document analysis and office automation department are presented. The activities serve as a basis for different document analysis projects. The results ascertained build the input for a message extraction and text understanding component which is described in a separate report [Baumann 95].

In order to present and to evaluate problem solving strategies and solutions, the overall text recognition task is divided into six subtasks, namely image preprocessing and layout extraction, font identification, character classification, voting of OCR results, and lexical postprocessing. The single problem solving strategies are combined to a global document analysis strategy by a reactive plan generation module. For each subtask possible solutions are discussed, and the current activities are presented in different detail.

Instead of a complete summary of the single activities we want to focus on the main characteristics of our planed and currently realized system.

Text Recognition Strategy: For text recognition there is no universal analysis strategy applicable to any document image. For this purpose, we have presented a plan-based analysis control mechanism activating specialized problems solvers in specific situations. Instead of a strict sequential analysis process, feedback loops between the single analysis phases can be realized. Additionally, the control mechanism proposed allows the distributed analysis where different analysis problems are solved using different workstations.

Layout Extraction for Mixed Mode Documents: For the image preprocessing and layout extraction task, a robust algorithm has been described. It is based on black and white connected components which are successively classified as text or nontext. Text components are grouped to character, word, line, and block segments and therefore serve as a good basis for the character and word recognition task. The approach presented has several advantages over other known techniques: It handles normal as well as inverse text, accepts characters in different font styles and sizes, is independent from text orientation, and allows the processing of characters which consist of more than one connected component.

Results Combination of Competitive Classifiers (Voting): To enable postprocessing on character hypotheses obtained by different OCR systems, we have developed a voting component. The goal is to combine results from different OCRs performing their tasks in parallel and to achieve higher recognition accuracy than any of the individual classifiers. To represent classification results on the measurement level and to improve classification accuracy, confusion matrices are learned in an offline learning phase. A voting architecture was proposed and first encouraging results with different representation and combination methods could be presented.

Lexical Postprocessing with Learning Capabilities: In the area of lexical postprocessing a flexible error correction mechanism by using knowledge about the allowed words, has been developed. It is based on an extension of the weighted edit distance, which closely reflects the distortions typically made by the layout extraction and character classification phase. Furthermore, a gradient descendant based learning algorithm for the edit costs was proposed. Experiments reveal its potential for this application domain. First theoretical results of the learning algorithm have been achieved, which prove its convergence property under the assumption that the rough form of the solution is already known.

6 References

[ALV 94] *ALV—Automatisches Lesen und Verstehen*; final project report; DFKI, April 1994.

[Appel 88] A. W. Appel, G. J. Jacobson. *The World´s fastest Scrabble Program*; in Com. of the ACM, Vol. 11, Number 5, 1988.

[Baird 90] H. S. Baird, S. E. Jones, S. J. Fortune. *Image Segmentation by Shape-Directed Covers*; in Proc. 10th ICPR, Atlantic City, USA 1990, Vol. II.

[Bartneck 89] N. Bartneck. *Methods for Photo Noise Extraction*; Daimler Benz Research Report, Ulm, 1989 (in German).

[Bartell 94] B.T. Bartell, G.W. Cottrell, R.K. Belew. *Automatic Combination of Multiple Ranked Retrieval Systems*; in Proc. of the 17th ACM-SIGIR 1994, Dublin, Ireland, July 1994.

[Batelle 91] *World Market Image Processing in Office Services*; Batelle Study, 1991.

[Baumann 95] S. Baumann, H.-G. Hein, R. Hoch, T. Kieninger, N. Kuhn, and M. Malburg. *Document Analysis at DFKI Part 2: Information Extraction*; Research Report at DFKI, 1995 (to appear).

[Bayer 94] T. Bayer, U. Bohnacker, and H. Mogg-Schneider. *InfoPortLab — an experimental document understanding system*; in Andreas Dengel and A. Lawrence Spitz, editors, IAPR Workshop on Document Analysis Systems, DAS 94, number D–94–06 in DFKI Document, pp. 297–312, Kaiserslautern, October 1994. IAPR.

[Bixler 88] J. P. Bixler. *Tracking Text in Mixed-Mode Documents*; Proc. Conf. on Document Processing Systems, Santa Fe, New Mexico, 1988.

[Black 63] D. Black. *The Theory of Comittees and Elections*; Cambridge University Press, London, England, 2nd ed., 1963.

[Bläsius 95] K.-H. Bläsius, F. Fein, I. John, N. Kuhn, and M. Malburg. *Document Analysis at DFKI Part 3: Knowledge Representation.* Research Report at DFKI, 1995 (to appear).

[Bradford 91] R. Bradford, T. Nartker. *Error Correlation in Contemporary OCR Systems*; in Proc. of the 1st ICDAR, Saint-Malo, France, 1991, pp. 516-524.

[Bunke 92] H. Bunke. *Recent Advances in String Matching*; in Advances in Structural and Syntactic Pattern Recognition, 1992, pp. 3-21.

[Bunke 93] H. Bunke, J. Csirik. *Inference of Edit Costs using parametric String Matching*; in Proc. 11th Int. Conf. on Pattern Rec., 1993, pp. 549-552.

[Chenevoy 91] Y. Chenevoy and A. Belaïd. *A hybrid approach for document image segmentation and encoding*; in [ICDAR 91], pp. 121–129.

[Chen 93] C.H. Chen and J.L. DeCurtins. *Word Recognition in a Segmentation-free Approach to OCR*; Proc. of the 2nd ICDAR, Japan, October 1993.

[Dasarathy 94] B.V. Dasarathy. *Decision Fusion*; IEEE Computer Society Press, Los Alamos, California, 1994.

[Dengel 92] A. Dengel. *ANASTASIL: A System for Low-Level and High-Level Geometric Analysis of Printed Documents*; in H. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*. Springer Publ., 1992.

[Diehl 91] Diehl. *Tame the Paper Tiger*; BYTE, special issue, vol. 16, no. 4, April 1991, pp. 220-

241.

[Duda 73] R. O. Duda, P. E. Hart. *Pattern classification and Scene Analysis*; John Wiley & Sons, 1973.

[Elliman 90] D. G. Elliman, I. T. Lancaster. *A Review of Segmentation and Contextual Analysis Techniques for Text Recognition*; Pattern Recognition, vol. 23, no. 3/4, 1990, pp. 337-346.

[Engelmore 88] R. S. Engelmore and A. J. Morgan, editors. *Blackboard Systems*; The Insight Series in Artificial Intelligence, Addison-Wesley Publishing Company, Wokingham, England, 1988.

[Esakov 94] Esakov, D. Lopresti, J. Sandberg. *Classification and distribution of optical character recognition errors;* SPIE Vol. 2181, Document Recognition, pp. 204-216, 1994.

[Fein 92] F. Fein and F. Hönes. *Model-based control strategy for document image analysis*; in Donald P. D'Amato, Wolf-Ekkehard Blanz, Byron E. Dom, and Sargur N. Srihari, editors, Machine Vision Applications in Character Recognition and Industrial Inspection, volume 1661 of *Proc. SPIE*, pages 247–256. SPIE — The International Society for Optical Engineering, February 1992.

[Fisher 90] J.L.Fisher, S.C.Hinds, D.P.D'Amato. *A Rule-Based System for Document Image Segmentation***;** in Proc. of the 10th ICPR, Atlantic City, USA (1990),pages 567–572.

[Fletcher 88] L.L.Fletcher, R.K.Kasturi. *A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images*; in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, 1988.

[Franke 92] J. Franke and E. Mandler. *A Comparison of Two Approaches for Combining the Votes of Cooperating Classifiers*; in Proc. 11th IAPR '92, The Hague, The Netherlands, 1992, pp. 611-614.

[Fu 86] K.-S. Fu. *Syntactic Pattern Recognition*; in Handbook of Pattern Recognition and Image Processing, ed. T. Y. Young and K.-S. Fu, 1986.

[Genesereth 88] M. R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*; Morgan Kaufmann Publishers, Palo Alto, CA, 1988.

[Hainzl 85] J.Hainzl. *Mathematik für Naturwissenschaftler*; Teubner Verlag, Stuttgart, 1985 (in German).

[Hall 80] P. A. Hall, G. R. Dowling. *Approximate Sting Matching*; Computer Surveys, Vol. 12, No. 4, pp. 381-402, 1980.

[Hayes-Roth 85] B. Hayes-Roth. *A blackboard architecture for control*; Artificial Intelligence, 26(3):251–321, 1985.

[Higashino 86] J.Higashino, H. Fujisawa, Y.Nakano, M.Ejiri. *A Knowledge-Based Segmentation Method for Document Understanding*; in Proc. 8th ICPR, Paris, France (1986).

[Hiltz 85] S.R. Hiltz and M. Turoff. *Structuring computer-mediated communication systems to avoid information overload*; Commun. ACM, vol. 7, no. 7 (1985), pp. 680-689.

[Ho 92a] T.K. Ho. *A Theory of Multiple Classifier Systems and Its Application to Visual Word Recognition*; technical report 92-12, Department of Computer Science, State University of New York at Buffalo, Buffalo, New York, May 1992.

[Ho 92b] T.K. Ho, J.J. Hull, S.N. Srihari. *On Multiple Classifier Systems for Pattern Recognition*;

in Proc. of the 11th IAPR, The Hague, The Netherlands, 1992, pp. 84-87.

[Ho 94] T. K. Ho, J. J. Hull, Sargur N. Srihari: *Decision Combination in Multiple Classifier Systems*; IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Vol. 16, No. 1, January 1994, pp. 66-75.

[Hoch93b] R. Hoch, T. Kieninger. *On virtual partitioning of large dictionaries for contextual post-processing to improve character recognition*; in Proc. of Second International Conference on Document Analysis and Recognition (ICDAR'93), Tsukuba Science City, Japan, October, 1993, pp. 226-231.

[Hönes 92] F.Hönes, R.Zimmer. *Separation of Textual and Non-Textual Information within Mixed-Mode Documents*; MVA'92, Tokyo, Japan (1992).

[Hönes 93] F. Hönes, J. Lichter. *Text String Extraction within Mixed-Mode Documents*; in Proc. Int. Conf. on Document Analysis and Recognition (ICDAR'93), Tsukuba Science City, Japan, 1993, pp.655-659.

[Hull 88] J. J. Hull. *A Computational Theory of Visual Word Recognition;* Technical Report 88-07 Department of Computer Science, State University of New York, 1988.

[Huang 93] Y.S. Huang and C.Y. Suen. *Combination of Multiple Classifiers with Measurement values*; in Proc of the 2nd ICDAR, Japan, October 1993, pp. 598-601.

[ICDAR 91] *Proceedings of the First International Conference on Document Analysis and Recognition, ICDAR '91*; Saint-Malo, France, September 1991. AFCET – IRISA/INRIA – Ecole Nationale Supérieure des Télécommunications.

[Kashyap 84] R. L. Kashyap, B. J. Oommen. *Spelling Correction using Probabilistic Method*; Pattern Recognition Letters 2, pp. 147-154, 1984.

[Kasper 94] T. Kasper. *Eine modellbasierte Steuerung für ein Dokumentanalysesystem*; Diplomarbeit, Universität Kaiserslautern, March 1994 (in German).

[Kreich 91] J. Kreich, A. Luhn, and G. Maderlechner. *An experimental environment for model based document analysis*; in [ICDAR 91], pages 50–58.

[Ling 89] X.Ling and W.G. Rudd. *Combining Opinions from Several Experts;* Applied Artificial Intelligence; Vol. 3, 1989, pp. 439-452.

[Lopresti 94] D. Lopresti, J. Zhou. *Using Consensus Sequence Voting to Correct OCR Errors*; in Proc. of the DAS 94, Kaiserslautern, Germany, October 1994, pp. 191-202.

[Kittler 86] J. Kittler. *Feature Selection and Extraction*; in Handbook of Pattern Recognition and Image Processing, ed. T. Y. Young and K.-S. Fu, 1986.

[Malburg 93] M. Malburg, A. Dengel. *Address Verification in structured documents for automatic mail delivery.* Proc. 1st European Conf. dedicated to Postal Technologies, vol. 1, SRTP Nantes, a1993,pp. 447-454.

[Mandler 88] E. Mandler and J. Schürmann. *Combining the Classification Results of Independent Classifiers Based on the Dempster/Shafer Theory of Evidence*; in Pattern Recognition and Artificial Intelligence, edited by E.S. Gelsema and L.N. Kanal, Elsevier Science Publishers B.V., North Holland, 1988, pp. 381-393.

[Mandler 90] E.Mandler, M.Oberlaender. *One-pass Encoding of Connected Components in Multi-Valued Images*; in Proc. 10th ICPR, Atlantic City, USA (1990).

[Marzal 93] A. Marzal, E. Vidal. *Computation of Normalized Edit Distances and Applications*; in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 9, September 1993, pp. 926-932.

[Nadler 84] M.Nadler. *Survey: Document Segmentation and Coding Techniques*; in Computer Vision. Graphics, and Image Processing, No. 28, 1984.

[Nagy 85] G.Nagy, S.Seth, S.D.Stoddard. *Document Analysis with an Expert System*; in Proc. Pattern Recognition in Practice II, Amsterdam, 1985.

[Nagy 92] G. Nagy. *Teaching a computer to read*; in Proc. 11th IAPR '92, The Hague, The Netherlands, 1992, pp. 225-229.

[Niyogi 86] D. Niyogi and S. N. Srihari. *A rule-based system for document understanding*; in AAAI-86, 5th National Conference on Artificial Intelligence, volume 2, pp. 789–793, Philadelphia, PA, August 1986. Morgan Kaufmann, Los Altos, CA.

[O'Gorman92] L.O'Gorman. *The Document Spectrum for Bottom-Up Page Layout Analysis*; in Proc. SSPR'92, Bern, Switzerland (1992).

[OMEGA 94] R. Bleisinger, S. Baumann, F. Fein, R. Hoch, F. Hönes, M. Malburg. *OMEGA: Office Mail Expert for Goad-Directed Analysis*; Project Proposal, DFKI, 1994.

[Peleg 79] S. Peleg. *Ambiguity Reduction in Handwriting with Ambiguous Segmentation and Uncertain Interpretation*; in Computer Graphics and Image Processing 10, pp. 235-245, 1979.

[Plessis 93] B. Plessis, A. Sicsu, L. Heutte, E. Menu, E. Lecolinet, O. Debon, J.-V. Moreau. *A Multi-Classifier Combination Strategy for the Recognition of Handwritten Cursive Words*; in Proc. of the 2nd ICDAR, Japan, October 1993, pp. 642-645.

[Rice 94] S. V. Rice, J. Kanai, T. A. Nartker. *The Third Annual Test of OCR Accuracy*; in 1994 Annual Report, UNLV Information Science Research Institute, pp. 11-39.

[Riseman 74] E. M. Riseman, A. R. Hanson. *A Contextual Postprocessing System for Error Correction Using Binary n-Grams*; in IEEE Trans. on Computers, Vol. c-23, No. 5, 1974, pp. 481-493.

[Randriamasy 94] S. Randriamasy. *A region-based system for the automatic evaluation of page segmentation algorithms*; in Proc. DAS'94, Kaiserslautern, Germany (1994).

[Sabourin 93] M. Sabourin, A Mitchie, D. Thomas, G. Nagy. *Classifier Combination for Hand-Printed Digit Recognition*; in Proc. of the 2nd ICDAR, Japan, October 1993, pp.163- 166.

[Schalkoff 92] R. Schalkoff. *Pattern Recognition Statistical, Structural and Neural Approaches*; 1992.

[Schürmann 77] J. Schürmann. *Polynomklassifikatoren für die Zeichenerkennung*; Oldenburg-Verlag, 1977 (in German).

[Schuhmann 87] M. Schuhmann. *Eingangspostbearbeitung in Bürokommunikationssystemen*; Springer Publ., 1987, (in German).

[Shafer 76] D. Shafer. *A Mathematical Theory of Evidence*; Princeton University Press, 1976.

[Shinghal 79] R. Shinghal, G. T. Toussaint. *Experiments in Text Recognition with the Modified Viterbi Algorithm*; in IEEE Trans. on Pattern Analysis and Machine Intell., Vol. PAMI-1, No. 2, 1979, pp. 184-192.

[Sinha 93] R. M. Sinha et al. *Hybrid Contextual Text Recognition with String Matching*; in IEEE

Trans. on Pattern Analysis and Mach. Intell., Vol. 15, No. 9, 1993, pp. 915-924.

[Srihari 1983] S. N. Srihari et al. *Integrating Diverse Knowledge Sources in Text Recognition*; in ACM Trans. on Office Information Systems, Vol. 1, No. 1, 1983, pp. 68-87.

[Suen 86] C. Y. Suen. *Character Recognition by Computer*; in Handbook of Pattern Recognition and Image Processing, ed. T. Y. Young and K.-S. Fu, 1986.

[Tahani 90] H. Tahani, J. M. Keller. I*nformation Fusion in Computer Vision Using the Fuzzy Integral; in* IEEE Trans. on Systems, Man, and Cybernetics, Vol. 20, No. 3, May/June 1990, pp. 733-741.

[Toussaint 78] G. T. Toussaint. *The Use of Context in Pattern Recognition*; in Pattern Recognition, Vol. 10, 1978, pp. 189-204.

[UNLV 93] UNLV Information Science Research Institute, Annual Report 1993.

[UNLV 94] UNLV Information Science Research Institute, Annual Report 1994.

[UW 93] *English Document Image Database I*; University of Washington, Seattle, WA, 1993.

[Wagner 74] R. A. Wagner, M. J. Fischer. *The String to String Correction Problem*; in Journal of the ACM, Vol. 21, No. 1, Jan. 74, pp. 168-173.

[Weigel 94] A. Weigel, S. Baumann. *A Modified Levenshtein Distance for Handwriting Recognition*; in Progress in Image Analysis and Processing III, ed. S. Impedovo, 1994.

[Weigel 94b] A. Weigel, F. Fein. *Normalizing the Weighted Edit Distance*; 12th Int. Conf. on Pattern Recognition, (Pattern Recognition and Neural Networks), 1994, pp. 399-402.

[Wahl 82] F.M.Wahl, K.Y.Wong, R.G.Casey. *Block Segmentation and Text Extraction in Mixed Text/Image Documents*; in Computer Graphics and Image Processing, Vol. 20, 1982.

[Wang 86] C.-H. Wang and S. N. Srihari. *Object recognition in structured and random environments: Locating address blocks on mail pieces;* in AAAI-86, 5th National Conference on Artificial Intelligence, volume 2, pp. 1133–1137, Philadelphia, PA, August 1986. Morgan Kaufmann, Los Altos, CA.

[Wang 89] D.Wang, S.N.Srihari. *Classification of Newspaper Image Blocks Using Texture Analysis*; in Computer Vision, Graphics, and Image Processing, Vol. 47, 1989.

[Yankelovich 85] N. Yankelovich, N. Meyrowitz. *Reading and Writing the Electronic Book; in* IEEE Computer, October 1985, pp. 15-30.

[Xu 92] L. Xu, A. Krzyzak, and C. Y. Suen. *Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition*; in IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 3, May/June 1992, pp. 418-435.

Appendix 1

The weighted edit distance is a well known measure to define the similarity of strings. It is used in a wide area of applications [Hall 80]. Usually the costs for the different edit operations are adjusted in an unsystematic way, resulting in fairly good weights. As an alternative we propose an iterative supervised learning algorithm, which adapts the weights based on a kind of gradient descendant approach. We show, that under some conditions the algorithm stops after a finite number of steps with a solution. Experiments reveal, that the weights computed by the learning algorithm, are very good solutions for typical classification problems.

1.1 Fundamentals

Let Σ be a finite alphabet and Σ^* be the set of all strings over Σ. According to the notation of [Wagner 74] and [Marzal 93], let $X = X_1X_2\ldots X_n$ be a string of Σ^*, where X_i is the *i*th symbol of *X*. We denote $X_{i\ldots j}$ the substring of *X*, which includes the symbols from X_i to X_j, $1 \le i, j \le n$. The length of such a string is defined as $|X_{i\ldots j}| = j - i + 1$. If $i > j$, $X_{i\ldots j}$ is the null string λ with $|\lambda| = 0$.

An edit operation is a pair $a \to b \neq \lambda \to \lambda$, where both *a* and *b* are strings of lengths 0 or 1. We restrict our focus on editing paths which are defined as follows. An editing path $P_{X,Y}$ between X and Y is a sequence of edit operations $X_{i_{k-1}+1\ldots i_k} \to Y_{j_{k-1}+1\ldots j_k}$, $1 \le k \le p$ satisfying the following:

$0 \le i_k \le |X|$, $0 \le j_k \le |Y|$,

$(i_0, j_0) = (0, 0)$, $(i_p, j_p) = (|X|, |Y|)$

$0 \le i_k - i_{k-1} \le 1$, $0 \le j_k - j_{k-1} \le 1$, $\forall k \ge 1$

$i_k - i_{k-1} + j_k - j_{k-1} \ge 1$

A weight function γ assigns to each edit operation a real number. According to [Weigel 94b] we allow positive and negative weights. The function γ can be extended to edit paths $P_{X,Y} = e_1e_2\ldots e_s$ by

$$\gamma(P_{X,Y}) = \sum_{i=1}^{s} \gamma(e_i) .$$

Assume $L = (eo_1, eo_2, \ldots, eo_m)$ is a list, which contains each possible edit operation exactly one time. We define a weight vector $G_{\gamma,L}$ for γ and L with $G_{\gamma,L}[i] = \gamma(eo_i)$. We further define a usage vector V for $P_{X,Y}$ and L by:

$V[i]$ = number of occurrences of eo_i in $P_{X,Y}$.

The *weighted edit distance ED(X, Y, γ)* between *X* and *Y* according to γ is defined as:

ED(X, Y, γ) = min$\{ \gamma(P_{X,Y}) \mid P_{X,Y}$ *is an editing path between X and Y*$\}$

$P_{X,Y}$ is called a minimal editing path between X, Y according to γ if $ED(X, Y, \gamma) = \gamma(P_{X,Y})$.

1.2 General learning algorithm

Properly setting of the costs for the elementary edit operations is a current research area [Bunke 92]. In [Bunke 93] a basis is set for an optimal determination of the costs. One main problem is the huge amount of time needed by this approach. In this section we propose an alternative learning

algorithm. It is based on an iterative learning approach with some kind of gradient descendant procedure [Duda 73].

We start with a formal description of the problem: Assume a set of strings $W = \{w_1, w_2, \ldots, w_n\}$ is given representing the different result classes. Furthermore, we have *n* string sets $B_i = \{b_1^i, b_2^i, \ldots b_{j_i}^i\}$, $1 \le i \le n$. The problem is to find a weight function γ such that

$$\forall b_v^u \in \bigcup_{i=1}^{n} B_i, \forall (w_t \neq w_u) : ED(b_v^u, w_u, \gamma) < ED(b_v^u, w_t, \gamma)$$

We propose some kind of gradient descendant method for solving this problem. Assume we start with some initial weights function γ_1 then the algorithm works as follows:

Algorithm 1

Input: $W = \{w_1, w_2, \ldots, w_n\}$

$B_i = \{b_1^i, b_2^i, \ldots b_{j_i}^i\}$, $1 \le i \le n$

initial weight function γ_1 ;

list of all edit operation $L = (eo_1, eo_2, \ldots, eo_m)$;

$\omega > 0$; // adaptation parameter

begin

i = 1;

while ($\exists b_v^u \in \bigcup_{i=1}^{n} B_i, \exists (w_t \neq w_u) : ED(b_v^u, w_u, \gamma_i) \ge ED(b_v^u, w_t, \gamma_i)$) do

for all $b_v^u \in B_1 \cup B_2 \ldots B_n$ do

if $\forall (w_t \neq w_u) : ED(b_v^u, w_u, \gamma_i) < ED(b_v^u, w_t, \gamma_i)$ do nothing

else

compute $w_r \neq w_u$ with $\forall (w_t \neq w_r) : ED((b_v^u, w_r, \gamma_i) \le ED(b_v^u, w_t, \gamma_i))$

compute a minimal editing path P_a between b_v^u and w_r according to γ_i

compute a minimal editing path P_b between b_v^u and w_u according to γ_i

build the usage Vector V_a for P_a and L

build the usage Vector V_b for P_b and L

for j = 1 to m do $\gamma_{i+1}(eo_j) = \gamma_i(eo_j) + \omega(V_a[j] - V_b[j])$ endfor

i = i + 1;

end else

end for

endwhile

return γ_i

end;

The algorithm realizes some kind of gradient descendant strategy. Each time a pattern b_v^u is mis-

classified, the weights are adapted in a way that the distance to the right string is reduced and the distance to the best fitting wrong one is enlarged.

It is obvious, that if Algorithm 1 terminates it returns a solution weight function. Unfortunately even if a solution weight function γ exists Algorithm 1 does not necessarily terminate. This is easily demonstrated with an example. Assume the following input data of Algorithm 1:

$W = \{w_1, w_2\}$, with $w_1 = b$ and $w_2 = c$,
$B_1 = \{b_1^1\}$, $B_2 = \{b_1^2\}$, with $b_1^1 = ah$ and $b_1^2 = a$
$\omega = 1$

A possible solution weight function is given by some γ with: $\gamma(a \to c) = 10$, $\gamma(a \to b) = 12$, $\gamma(h \to b) = 5$, $\gamma(a \to \lambda) = 10$, and for all other edit operations the costs are equal to 1000. If we start Algorithm 1 with an initial weight function γ_1 defined by: $\gamma_1(a \to b) = 1$, $\gamma_1(a \to c) = 2$, and for all other edit operations the costs are equal to 1000, then it computes an infinite chain of weight functions γ_2 , γ_3 ,... with: $\gamma_k(a \to b) = 1$, $\gamma_k(a \to c) = 2$ for all odd k and $\gamma_k(a \to b) = 2$, $\gamma_k(a \to c) = 1$ for all even k.

1.3 Conditions for convergation

In this section we want to discuss a slight modification of Algorithm 1, using information about the rough form of the solution, which always computes a solution weight function if such a function exists. Again we have a set of strings $W = \{w_1, w_2, \ldots, w_n\}$ representing the different result classes. Furthermore, we have n string sets $B_i = \{b_1^i, b_2^i, \ldots b_{j_i}^i\}$, $1 \le i \le n$. As before the problem is to find a weight function γ such that

$$\forall b_v^u \in \bigcup_{i=1}^{n} B_i, \forall (w_t \neq w_u) : ED((b_v^u, w_u, \gamma) < ED(b_v^u, w_t, \gamma))$$

Given some initial weight function γ_1 the problem is that for some b_v^u there exists a $w_t \neq w_u$ with $ED(b_v^u, w_t, \gamma) \le ED(b_v^u, w_u, \gamma)$. Nevertheless we often know the minimal editing path $P(b_v^u, w_u)$ between b_v^u and w_u according to the solution vector γ (which we do not know at this moment). Usually $P(b_v^u, w_u)$ reflects the distortions which have transformed w_u into b_v^u. $P(b_v^u, w_u)$ can be easily constructed by matching both strings with a rough approximation γ_ε of γ. In spite of some uncertainties a user can decide which editing path reflects best the distortions. Formally, we have the situation in which we know for all $b_v^u \in \bigcup_{i=1}^{n} B_i$ a minimal path $P_\varepsilon(b_v^u, w_u)$ between b_v^u and w_u according to the unknown solution vector γ. As an example look at the strings *aab* and *aac*. It is reasonable to assume, that the minimal editing path according to γ is $a \to a, a \to a, b \to c$. Note that we actually do not know the solution weight function γ itself.

Algorithm 2 uses the knowledge about the right form of the minimal solution edit paths to achieve its goal, i. e. a solution weight function.

Algorithm 2

Input:

$W = \{w_1, w_2, \ldots, w_n\}$

$B_i = \{b_1^i, b_2^i, \ldots b_{j_i}^i\}$, $1 \le i \le n$

initial weight function γ_1 ;

list of all edit operation $L = (eo_1, eo_2, \ldots, eo_m)$;

$\forall u, v$ a minimal editing path $P_\varepsilon(b_v^u, w_u)$ according to an unknown solution weight function γ[34]

$\omega > 0$;

begin

i = 1;

while ($\exists b_v^u \in \bigcup_{i=1}^{n} B_i, \exists (w_t \neq w_u) : ED(b_v^u, w_u, \gamma_i) \ge ED(b_v^u, w_t, \gamma_i)$) do

for all $b_v^u \in \bigcup_{i=1}^{n} B_i$ do

if $\forall w_t \neq w_u : ED((b_v^u, w_u, \gamma_i) \le ED(b_v^u, w_t, \gamma_i))$ do nothing

else

compute $w_r \neq w_u$ with $\forall w_t \neq w_r : ED((b_v^u, w_r, \gamma_i) \le ED(b_v^u, w_t, \gamma_i))$

compute a minimal editing path P_a between b_v^u and w_r according to γ_i

build the usage Vector V_a^i for P_a and L

build the usage Vector V_b^i for $P_\varepsilon(b_v^u, w_u)$ and L

for j = 1 to m do $\gamma_{i+1}(eo_j) = \gamma_i(eo_j) + \omega(V_a^i[j] - V_b^i[j])$ endfor

i = i + 1;

end else

end for

endwhile

return γ_i

end;

Algorithm 2 stops with a solution vector γ_S if a solution weight function γ exists. This property will be proofed next. The proof is strongly oriented at the proof of the convergence of the minimization of the Perceptron Criterion Function [Duda 73].

34. The unknown solution weight function must be fixed for all minimal editing paths. Note that there may be several solution weight functions, which can all be used as an attractor.

Proof:

Let $G_{\gamma_i, L}$ be the weight vector for γ_i and L and $G_{\gamma, L}$ the weight vector for γ and L. Then it holds

$$(G_{\gamma_{i+1}, L} - \alpha G_{\gamma, L}) = (G_{\gamma_i, L} - \alpha G_{\gamma, L}) + \omega (V_a^i - V_b^i)$$

and hence

$$\left\|G_{\gamma_{i+1}, L} - \alpha G_{\gamma, L}\right\|^2 = \left\|G_{\gamma_i, L} - \alpha G_{\gamma, L}\right\|^2 + 2\omega\Big((G_{\gamma_i, L} - \alpha G_{\gamma, L})^t (V_a^i - V_b^i) \Big) + \left\|\omega (V_a^i - V_b^i)\right\|^2,$$

where A^t denotes the transpose of *A*.

We know that $G_{\gamma_i, L}{}^t (V_a^i - V_b^i) \le 0$, because the distance to the right sample is larger than the distance to the best wrong one and thus

$$\left\|G_{\gamma_{i+1}, L} - \alpha G_{\gamma, L}\right\|^2 \le \left\|G_{\gamma_i, L} - \alpha G_{\gamma, L}\right\|^2 - 2\alpha\omega G_{\gamma, L}{}^t (V_a^i - V_b^i) + \left\|\omega (V_a^i - V_b^i)\right\|^2$$

What we want to show is that there exists an α such that

$$-2\alpha\omega G_{\gamma, L}{}^t (V_a^i - V_b^i) + \left\|\omega (V_a^i - V_b^i)\right\|^2 < -1$$

$$\Leftrightarrow -2\alpha\omega G_{\gamma, L}{}^t (V_a^i - V_b^i) < -1 - \left\|\omega (V_a^i - V_b^i)\right\|^2$$

$$\Leftrightarrow 2\alpha\omega G_{\gamma, L}{}^t (V_a^i - V_b^i) > 1 + \left\|\omega (V_a^i - V_b^i)\right\|^2$$

$$\Leftrightarrow \alpha > \frac{1 + \left\|\omega (V_a^i - V_b^i)\right\|^2}{2\omega G_{\gamma, L}{}^t (V_a^i - V_b^i)}$$

There exists $\beta > 0$ with $\beta > V_a^j[k] \ge 0$ and $\beta > V_b^p[q] \ge 0$ for all j, k, p and q, because the number how often a single edit operation can be applied for a match between two of the involved strings is restricted by the definition of editing paths. Therefore it exists a $\mu > 0$ with $\mu > \left\|\omega (V_a^k - V_b^k)\right\|^2 \ge 0$ for all k.

Furthermore it holds $G_{\gamma, L}{}^t (V_a^j - V_b^j) > 0$ [35].

Remember $\beta > V_a^j[k] \ge 0$ and $\beta > V_b^p[q] \ge 0$ for all j, k, p, and q. Furthermore, each $V_a^j([k])$ and $V_b^p[q]$ must be an integer. Therefore, there is only a finite number of possible vectors $V_a^j - V_b^j$ and a finite number of vectors $V_a^j - V_b^j$ with $G_{\gamma, L}{}^t (V_a^j - V_b^j) > 0$. From this it follows that there must be an η with $2\omega G_{\gamma, L}{}^t (V_a^j - V_b^j) > \eta > 0$.

$\Rightarrow \alpha > \frac{1 + \mu}{\eta}$ is a solution for $-\alpha\omega G_{\gamma, L}{}^t (V_a^i - V_b^i) + \left\|\omega (V_a^i - V_b^i)\right\|^2 < -1$

$\Rightarrow \left\|G_{\gamma_{i+1}, L} - \alpha G_{\gamma, L}\right\|^2 \le \left\|G_{\gamma_i, L} - \alpha G_{\gamma, L}\right\|^2 - 1$

Because a squared distance can never become negative it follows that there can't be an infinite sequence of corrections. Therefore the corrections must terminate with a solution weight function.

Q.E.D.

1.4 Experiments

In this section we describe an experiment which demonstrates the usefulness of our learning algorithm for lexical postprocessing of OCR results. Assume we want to recognize words, whereby the vocabulary is represented by the string set $W = \{w_1, w_2, \ldots, w_n\}$. For training we have *n* string sets $B_i = \{b_1^i, b_2^i, \ldots b_{j_i}^i\}$, $1 \le i \le n$, whereby each b_j^i is the output of the OCR devise for an input of the word w_i . The training strings are used to infer a weight function γ with Algorithm 1, which hopefully reflects the typical distortion made during the recognition task. Furthermore we have *n*

35. Exactly this condition is not fulfilled in the section before.

independent test sets $T_i = \{t_1^i, t_2^i, \ldots t_{h_i}^i\}$, $1 \le i \le n$, whereby each t_j^i is the output of the OCR devise for an input of the word w_i. To test if our learning algorithm was successful, we count the number of misrecognized strings CR, defined by

$$CR = \left| \{ t_v^u \,|\, \exists\, (w_t \neq w_u), ED\,(t_v^u, w_u, \gamma) \ge ED\,(t_v^u, w_t, \gamma) \} \right| .$$

Because we can not be sure that there exists a solution weight function correctly separating the training sets, we have to modify the condition of the while loop in Algorithm 1. Now we use a counter *count* which defines how often the while-loop should be executed.

In our experiments we used a document database of about 90 scanned business letters many of them of bad image quality. For all documents the recognition results are stored. Together the documents contain about 9500 words. The document set is divided into a training set containing about 8300 words and a test set containing about 1200 words. The training starts with some initial values for the edit costs, carefully selected during experiments, but with the restriction, that edit operations of the same type (deletion, substitution etc.) do have the same costs.

Training of the edit costs was done by running through the train set for 20 times, i. e. *count* is set equal to 20. After each traversal the intermediate values for the edit costs are tested against the test set. In Figure 17 the number of misrecognized words CR in the test set after each training cycle is shown, starting with about 120 and decreasing to 70.

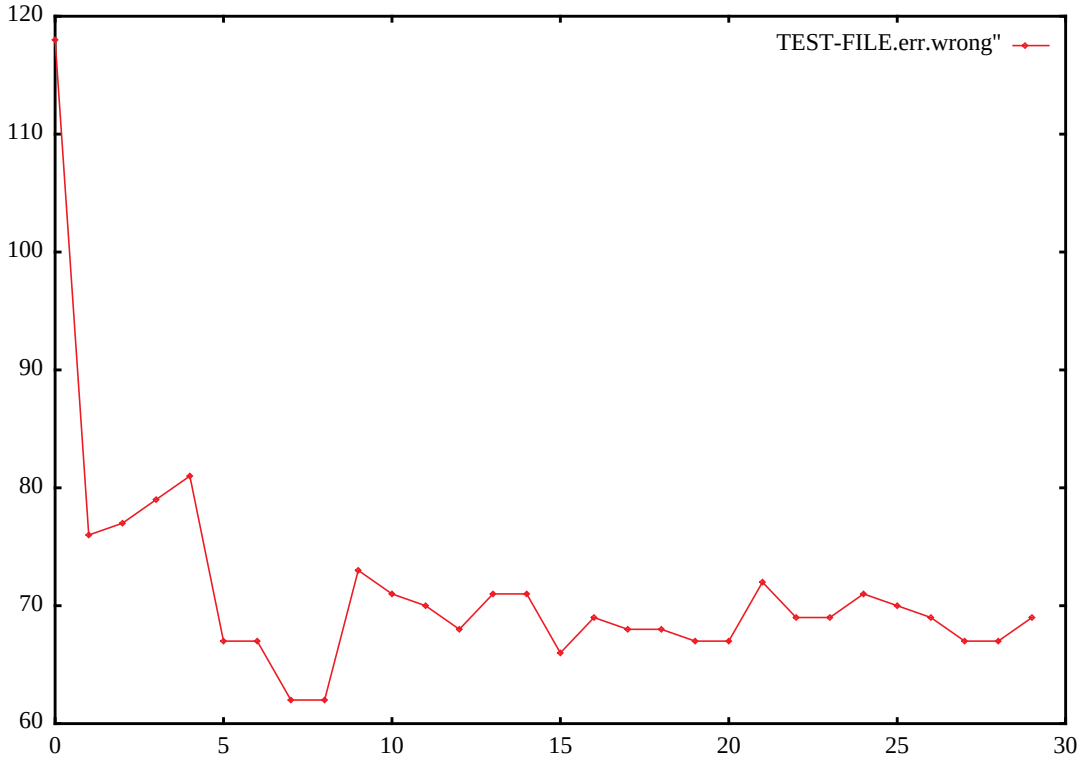

Figure 21: Wrong words

The improvements of the recognition results due to the training procedure are obvious. Only about 7 training steps are necessary to achieve the best results. We think further progress is possible by building font specific postprocessing modules, which models more adequate the specific errors of different document font types. Also the integration of additional edit operations for merges and split ups will improve the recognition accuracy as well as the efficiency of the learning algorithm.

1.5 Conclusion

In this Appendix we discussed some properties of an iterative learning algorithm for the costs of the elementary edit operations of the weighted edit distance. Unfortunately, in its general form this algorithm does not find a solution even if the training set is separable. Under the assumption that at least the rough form of the solution is known, a slight modification of this approach terminates with a solution, if one exists.

Experiments revealed, that even in the inseparable case impressive results can be achieved by this learning strategy.