THE APPLICATION OF WORD IMAGE MATCHING IN TEXT RECOGNITION

by

Siamak Khoubyari

A Thesis
Submitted to the Faculty of the Graduate School
of State University of New York at Buffalo
in partial fulfillment of the requirements
for the degree of Master of Science

June 1992

Committee Members

Chairman:	Dr. Jonathan J. Hull Computer Science, SUNY at Buffalo
Member:	Dr. Sargur N. Srihari Computer Science, SUNY at Buffalo

This thesis was defended on April 29, 1992.

To my family:
whose love and support
made my education possible

Acknowledgments

My sincere gratitude goes to the following people:

Dr. Jonathan J. Hull, my advisor, who helped me with every stage of my research. His inspiring dedication, and thoughtful advice will always be appreciated.

Dr. Sargur N. Srihari, for his insight and helpful comments and encouragement.

Dr. Tin Kam Ho, who was always willing to assist me with my questions and provide valuable comments.

Mr. Scott Cohen, for his technical support and encouraging conversations.

Contributions by Dr. Hull and Dr. Ho have been used in this work.

Special thanks to Jackie Kud, who provided encouragement and moral support.

Thanks also to the secretaries, especially Eloise Benzel, Heike Berlind, Sally Elder, Eugenia Smith and Debora Squire. They helped me find my way through the bureaucracy.

Abstract

Printed text recognition usually involves recognizing individual characters, and assembling the results to recognize words and sentences. However, the performance of conventional character recognition systems tends to suffer in the presence of moderate levels of degradation in the text.

A method is proposed that uses equivalence among frequent *word* images to derive hypotheses for each word using the available language statistics. Word images are clustered to determine equivalency. The attributes of the clusters, as well as the relationships among them are then matched with the same characteristics for the words in the language. The method requires no explicit training, and is fairly tolerant to image degradation. The results for several sample sizes are reported.

Contents

1 Introduction **4**
- 1.1 Motivation . . . 4
- 1.2 Problem Definition . . . 5
- 1.3 This Thesis . . . 6

2 Background **7**
- 2.1 The OCR Process . . . 7
- 2.2 Isolated Character Recognition . . . 8
  - 2.2.1 Template Matching . . . 8
  - 2.2.2 Other Global Methods . . . 9
  - 2.2.3 Feature Analysis . . . 11
- 2.3 Word Recognition . . . 12
  - 2.3.1 Character-Based Methods . . . 13
  - 2.3.2 Segmentation-Based Methods . . . 14
  - 2.3.3 Holistic Word Recognition . . . 15
- 2.4 Discussion . . . 17

3 Proposed Solution **22**
- 3.1 Word Image Matching by Clustering . . . 22
  - 3.1.1 Introduction . . . 22
  - 3.1.2 The Role of Constraints . . . 24
- 3.2 Cluster Identification . . . 27
  - 3.2.1 General Approach . . . 27
  - 3.2.2 Matching Techniques . . . 30
- 3.3 Discussion . . . 32

4 Details of the Solution **35**
- 4.1 Clustering . . . 35
  - 4.1.1 Cluster Initialization . . . 36
  - 4.1.2 Finding the Closest Cluster Pair . . . 37
  - 4.1.3 Merging Clusters . . . 40

4.2 General Word Matching 41
4.2.1 Introduction 41
4.2.2 Word Autocorrelation 41

5 Empirical Study **47**
5.1 Experimental Environment 47
5.1.1 The Brown Corpus 47
5.1.2 Automatic Text Image Generation 49
5.1.3 Introducing Noise 52
5.2 Clustering Results 56
5.2.1 Error Analysis 56
5.2.2 Results 58
5.2.3 Alternative Similarity Measure 66
5.3 Cluster Identification 67

6 Conclusions and Future Research **72**

Bibliography **74**

A Brown Corpus Markers **78**
A.1 Graphics Markers 78
A.2 Syntactic Tags 78

B Clustering Output **81**

Chapter 1

Introduction

Even with the many advances in technology, humans still tend to prefer the paper communication medium to the electronic one. Printed text is used in every aspect of life as a tool for the transfer of knowledge and information. On the other hand, computers allow for the quick and efficient use of large on-line databases, simplifying many *information retrieval* tasks. Still, entering printed material into a computer using the usual input devices, such as keyboards, is tedious, time-consuming, and highly inefficient, considering the high volume of printed text that is produced every day. The purpose of printed text recognition research has been to bridge the gap between the printed and electronic media.

1.1 Motivation

Giving computers the ability to automatically “read” printed documents, and convert them to an on-line format (e.g. ASCII) is a challenging pattern recognition problem. Text recognition as a research problem has attracted many scientists and engineers over the past four decades for this reason.

Much of the information used daily for various purposes is gathered from large volumes of printed text. Furthermore, much of the data is manually entered into computers by human operators using keyboards, facilitating electronic storage and future retrieval or processing of the information. Unfortunately, the rate of data entry is limited by conventional methods, even as the processing speed of computers increases.

From an economic standpoint, fast and efficient automation of data entry from printed sources would be beneficial. By eliminating the data entry "bottleneck", human operators can attend to more important tasks, while the computers can more efficiently process the incoming data.

The use of Optical Character Recognition (OCR) systems has been investigated for many different applications. These are usually, but not exclusively, applications where a large number of data items need to be processed. Perhaps the most obvious use of OCRs is in the construction of large on-line libraries, where printed material such as books, journals, reports, and so on are electronically stored and processed.

Studying the abstract problem of "reading" from an AI point of view is also interesting. Reading is a natural process for most people who have been introduced to the basics. However, teaching computers to "read" written material as easily and efficiently as humans do is a yet unattained goal. Therefore, as a research problem, developing such a machine is a challenge. Furthermore, the performance of current OCR systems is still worse than a typical human reader, leaving much room for improvement. Hence, the focus of much of the research in this area has been to develop new techniques to be used in reading machines, *possibly* making the OCRs of tomorrow as accurate and reliable as human readers.

1.2 Problem Definition

This thesis considers the problem of automatic reading of printed text by a computer system. The goal is to develop a system with performance similar to that of human readers. The performance of current OCR methods tends to decrease when the text used is moderately degraded, with broken characters and/or characters that are touching. Therefore, new methods must be devised that possess the robustness required when dealing with low-quality printed text. It would seem obvious that such a method should consider not only the physical appearance of the text, but also alternative sources of information, such as information about the characteristics of the language.

1.3 This Thesis

Chapter 2 gives a brief description of the OCR process, followed by description of some recognition methods that fall under two general categories: *character recognition* and *word recognition*. Some well-known techniques in each category are briefly described. A comparative discussion of the various methods concludes the chapter.

In chapter 3, the proposed solution is outlined, giving a general overview of the different components. The use of *word image matching* and *clustering* is explained. The significance of each component, as well as the merits of the solution are also discussed. This is followed, in chapter 4, by a detailed description of the various algorithms used in implementing the system.

Chapter 5 discusses the experimental environment in some detail. The experimental results are also reported and discussed. The thesis is concluded in chapter 6, where a summary of the solution and future considerations are presented.

Chapter 2

Background

2.1 The OCR Process

Although optical character recognition (OCR) algorithms are very diverse in their details, the basic process is fairly similar for all of them. The main components of an OCR system are illustrated in Figure 2.1, with the shaded boxes representing steps that may only pertain to certain classes of algorithms.

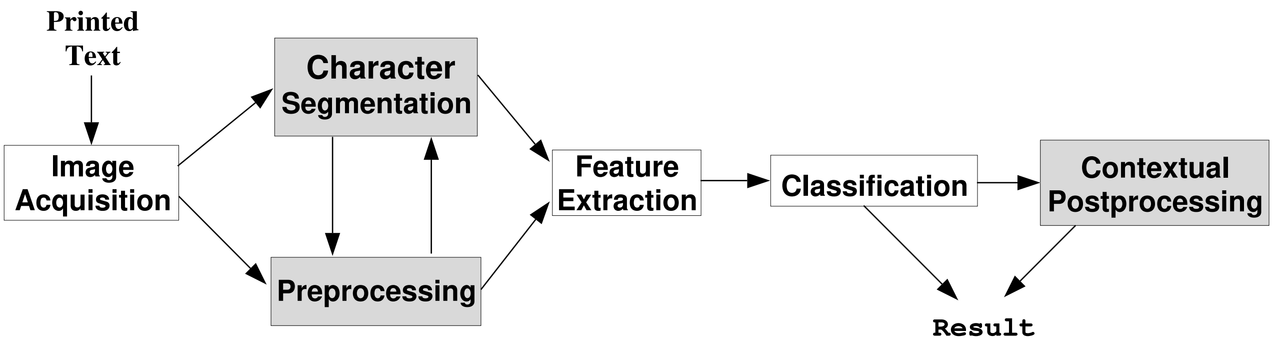


Figure 2.1: General steps in most OCR algorithms.

With a few exceptions, image acquisition is mainly performed by scanning a page of text using an optical scanner, and transforming it to an on-line representation using some encoding mechanism. For example, a *grey-scale* image of text may be presented by assigning each pixel a number (say, between 0 to 255) corresponding to its shade of grey. A *binary* image would then contain only two pixel values: 0 for black and 255 for white. Once the image is scanned, it may be passed through a segmentation step, where a page of text may be segmented into words, and further, the words themselves may be segmented into the component characters. The segmented word or character images can then be passed through a series of preprocessing filters, depending on the algorithm used. The purpose of preprocessing is to improve image quality, or to facilitate feature extraction.

Feature extraction is an integral part of almost all OCR algorithms. It is in this step that distinguishing characteristics are gathered from the input image and organized so that the particular recognition algorithm can make good use of them. The recognition (or *classification*) step can then use the feature descriptions and determine the identity of the unknown character or word. Finally, some OCR algorithms contain a postprocessing step in which different information are used to check the derived identity, and make necessary corrections to the output.

Many approaches to the problem of text recognition have been investigated, and have been outlined in [15] [32]. The following two sections provide a brief summary of some important techniques in two broad categories: *isolated character recognition* and *word recognition*. The chapter will be concluded with a discussion of the various techniques.

2.2 Isolated Character Recognition

2.2.1 Template Matching

Template Matching is an intuitive character recognition technique used since the earliest OCRs. This method involves comparing global information about a character image (e.g. the value of the pixels or the shape of the character's boundary) with the same information from a known set of patterns (character **templates**).

In some early OCRs [14] [37], the physical *optical* image of the segmented character was compared with a series of pre-defined masks (templates) representing the alphabet. By super-imposing the character image over all the masks, the best match was found and the unknown character was identified.

In later systems, digitized *bitmaps* (pixel representations) of the images were used to represent the input character as well as the template patterns [33]. A simple correlation method, such as comparing each bit (pixel) in the unknown character to the corresponding bit in the template, was then used to find the best match to the input.

Other variations of the template matching technique in character recognition have since been developed with the goal of increasing the accuracy of the recognition method. Casey and Nagy use a "decision tree" to perform step-wise discrimination among character patterns to ultimately recognize them [7]. At each node of the tree, a binary decision is made on the basis of the pixel value in a particular region of the image. These decisions are made hierarchically until a unique choice is identified. This type of refinement makes the classification of patterns simpler and more accurate by focusing the decision-making on a unique sub-problem at each step, and hence reducing the number of classes associated with each level of the tree. The accuracy of this method can be further improved by using multiple decision trees for each class of characters in a particular font [5].

2.2.2 Other Global Methods

Transformations

One of the disadvantages of the template matching technique is its high sensitivity to distortions in the image. The random noise, and variations in the orientation can change the appearance of the input in a way that it would be difficult to find an accurate match to a template.

Using a transformation such as the *Fourier* method, more general image descriptions can be obtained, which are not affected as much by common pattern distortions. Brill [4] and Granlund [13] use information obtained from the contour of the input character (called the *Characteristic Waveform*), and expand it into a Fourier series. The resulting Fourier coefficients are then cross-correlated with the same information from a “reference set” of characters. This part of the method is similar to template matching, but the detailed information reflecting the orientation, scaling, degradation, etc. are filtered out, making the recognition of slightly noisy images more feasible. It is also shown that at least in the case of handwritten digits, using only 10 to 15 Fourier coefficients can give a full description of the contour of the input image, dramatically decreasing the amount of data needed to perform the recognition [4].

Distribution Analysis

In classic template matching methods, each pixel in the character image is used in determining the identity of the characters. An alternative approach is to use the pixels’ statistical distribution in the classification process.

One such method uses many “n-tuples” to examine and classify the distribution of the pixels in the image. N-tuples are $n \times n$ arrangements of certain pixel distributions, usually describing a significant “micro” pattern. Bakis, et. al. [2] use both manually and computer-generated n-tuples in the recognition of hand-printed numerals. They also use overlapping regions in the image, called “zones”, which may contain different orientations of the n-tuples. Using such detailed analysis of the image, an accurate description of the pixel distribution in the pattern is extracted. Also, Bakis has demonstrated an increase in the recognition performance when multiple computer-generated n-tuples are used in conjunction with manually-generated ones.

Another related method uses geometrical moments to classify patterns. Moments are calculated in several orders, describing the distribution of black pixels in the image. These are then compared with the pre-calculated values obtained from a *training set*. This method is generally used in conjunction with other recognition techniques to arrive at more reliable decisions [28] [36].

2.2.3 Feature Analysis

The most popular class of algorithms in character recognition deal with the use of a particular set of discriminating characteristics (called *features*) to distinguish among characters. Methods using many different types of features ranging from simple to very complex patterns have been discussed in the literature. The ideal set of features would be simple, computationally inexpensive, and yet fairly invariant to normal distortions effecting character images.

Many of the features used in character recognition attempt to exploit geometrical characteristics of the input pattern in some way. Zazmierczak [21] uses a few of these geometrical shapes to classify character images. The character pattern is divided into regions, and certain geometrical shapes such as *peaks, line endings, curves* and *bays* are detected in each region. The binary results for all the features are then grouped into a feature vector and compared with the pre-defined "shape criteria" for each class of characters.

Variations of this method have been developed by many researchers. Sublette and Tults [30] use a total of 56 geometrical features categorizing vertical lines, edge properties, white regions, etc. To extract some of these features, various "edge-tracing" algorithms are used.

Tou and Gonzalez [34] use features in a multi-level classification scheme. Here, the recognition is performed in multiple stages according to a pre-defined hierarchy. First, global geometrical features are extracted from the characters image

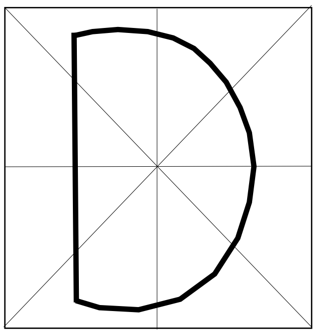


Figure 2.2: A measurement grid used for feature extraction

in each of the cells in an octagonal grid (figure 2.2). Using the resulting feature vector, each input character is categorized into one of several subsets of the alphabet. The second stage attempts to extract more detailed, local features from the unknown character, placing it in a smaller (possibly singleton) subset. If more recognition is needed, specific features distinguishing the individual members of the latter subset are extracted, giving enough information for unique classification. Of course, the classes used in the such hierarchies are pre-determined, as are the individual character descriptions, using prior information collected from a preferably large training set. A similar method using only a *bi-level* strategy was developed by Spinrad [29].

2.3 Word Recognition

The techniques mentioned in the previous section generally deal with deriving the identity of isolated characters and are not concerned with recognizing entire words. Some algorithms for word recognition will be discussed in this section.

In general, character recognition methods use only the information obtained from the isolated character pattern itself to identify it. This is possible because, when dealing with characters, only a small number of different classes, i.e. different characters, need to be distinguished. Conversely, word recognition methods cannot assume such a small set of possible patterns, since it is impractical to attempt to represent *all* word classes. For this reason, word recognition methods must use other *contextual* information. The simplest source of such information is a *dictionary*, or *lexicon* of all or many of the possible words that may appear in documents pertaining to a particular application.

Algorithms that aim at identifying words can be characterized by how much, and what level of contextual knowledge they use. On this basis, Ho, Hull and Srihari [17] classify word recognition technique into *character-based*, *segmentation-based* and *word shape* or *holistic* methods.

2.3.1 Character-Based Methods

The character-based methods identify a word by recognizing each character sequentially. Since errors can occur at the character level, the result from the recognition process must be *postprocessed* to correct "spelling" mistakes. This usually entails the use of a dictionary to determine the correctness of the results, and then using other sources of knowledge, such as the identity of adjacent characters, to correct any errors. Figure 2.3 shows this process pictorially.

Since character-based methods rely on isolated character recognition results to make their decision, they are usually suitable for applications in which characters are easily segmentable and relatively free of distortions, making character recognition feasible [17].

Several methods can be used for post-processing words that have been formed by isolated character recognition. Direct use of a dictionary is the most obvious and straightforward of these. A search is made to locate the recognized word inside the dictionary; if no matches can be found, the dictionary word that is closest to the unknown word is picked as the correct answer.

Although this intuitive approach may be suitable for applications which require only a small lexicon, it becomes increasingly expensive as the size of the dictionary grows. Therefore, other methods have been developed to abstract or compact the information contained in the dictionary.

Riseman and Hanson [27] use inter-character relationships within words to

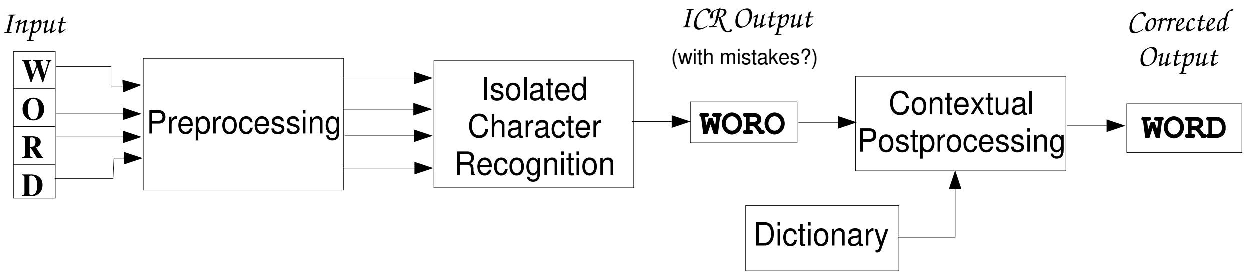


Figure 2.3: Character-based Word Recognition with Contextual Postprocessing

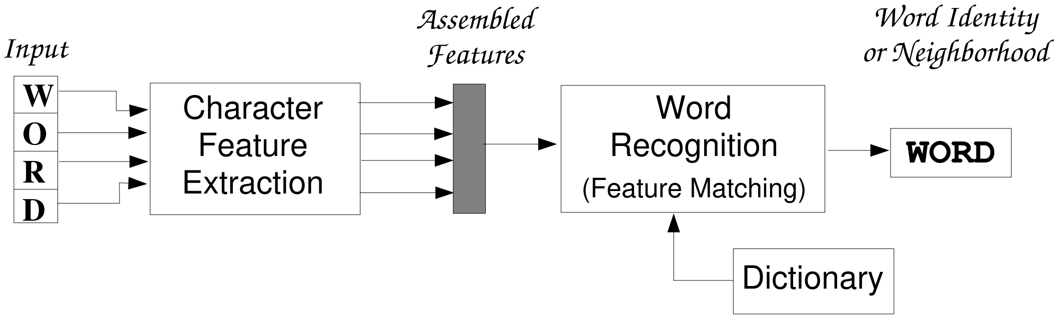


Figure 2.4: Segmentation-based Word Recognition

capture the information in the dictionary and to correct the results of isolated character recognition. Specifically, they constrain the combinations of letter pairs or triplets within individual words using corresponding *a priori* probabilities called *bigrams* and *trigrams*. Each letter combination that occurs in the dictionary words has a probability which can be used to increase or decrease the likelihood of that pair having occurred in the unknown word.

Character-based methods have also been developed *without* the use of a dictionary. Downton, et. al. [9] use character-based word recognition to identify postal codes on mail pieces. The character recognition decisions are postprocessed using knowledge about the format of the postcodes, as well as information gathered from the rest of the mail piece. However, they do suggest that to enhance performance, (postcode) dictionary information should be included in the system.

2.3.2 Segmentation-Based Methods

Segmentation-based methods are similar to Character-based methods because they make use of isolated character recognition results. However, they do not use the derived identities of the characters to compose words. Rather, these methods compile a description of the word by appending the *feature* descriptions of the isolated characters. This feature vector can be combined with contextual knowledge to locate the most similar dictionary word (figure 2.4).

Segmentation-based techniques are suitable for words that contain somewhat degraded, but easily segmentable characters. This is because even if the character recognition step is robust enough to derive an accurate description of noisy characters, it would be difficult to recognize the word if the characters are not segmented correctly.

An example segmentation-based method was developed by Duderstadt, et. al. [10], who use a set of features to describe each of the characters, and derive a *feature vector* for the word by combining the individual character features. However, they must use multiple segmentation *alternatives* to account for the possibility of incorrect segmentation of words. A set of topological features are extracted from each of the segmented "symbols", and the values from each type of feature are put together to create a feature vector for the whole word. A word lookup step follows which searches through a lexicon of words in four fonts. The comparison between the unknown word's feature description and the lexicon words is performed using a simple correlation method.

2.3.3 Holistic Word Recognition

As mentioned earlier, word recognition methods differ in how much, and what levels of contextual knowledge they use to perform their task. The character-based methods described above use context in a *bottom-up* manner. That is, the local character-level information is used to identify the words. An alternative approach is to use global information, such as descriptions of whole words to recognize the word images. These techniques can be particularly useful if the word images are noisy

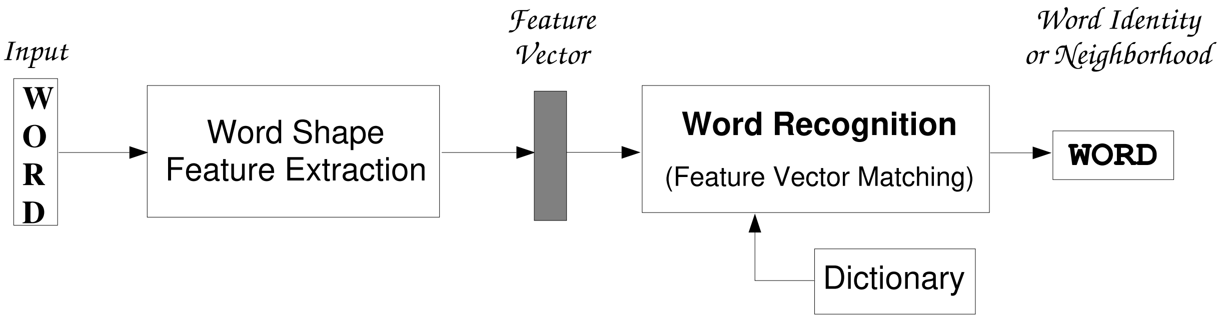


Figure 2.5: A holistic word recognition system

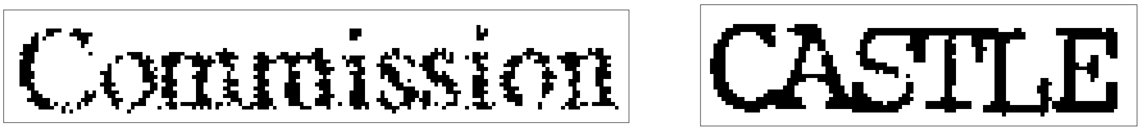


Figure 2.6: Examples of word images suitable for holistic word recognition

and/or very difficult to segment (figure 2.6), since these kinds of images are generally difficult to recognize using the other two word recognition methods.

Holistic, or *word shape* analysis uses features that are derived from the shape of the whole word, rather than from segmented symbols. The word's feature description is then matched to the ones for dictionary words, resulting in the word's identity, or a small subset of dictionary words (a *neighborhood*) that best match the unknown. Other postprocessing steps can then choose the correct word, among those in the neighborhood, using alternative knowledge sources.

One of the first recognition techniques that used word attributes was presented by Earnest [11], in a system for *handwriting* recognition. Each word is partitioned into 3 regions (ascender, middle and descender), and significant features are extracted from each region (see figure 2.7), yielding a feature vector for the word. Each category of features is then used separately to incrementally partition a dictionary of more than 10,000 words into smaller and smaller subsets, until the correct word is chosen.

Brickman and Rosenbaum [3] use a dictionary of the 150 most common word in English, representing about 50% of a normal written documents, as their only external knowledge source other than global word descriptions. They first use some word characteristics (the presence and location of ascenders and descenders) to narrow the list of possible matches. A simple word autocorrelation technique is then used to find the dictionary word that best matches the unknown. Even though this simple method can not achieve high rates of recognition, partly due to the small dictionary size, it is suitable for the particular application that it was used in, namely,

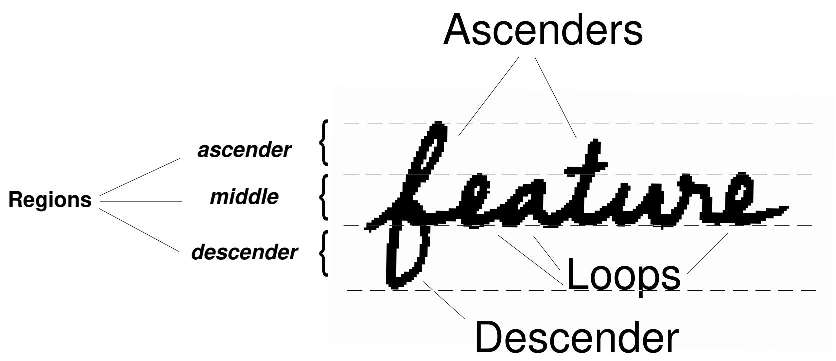


Figure 2.7: Feature extraction using word reference lines

facsimile image compression.

Ho, Hull and Srihari [16] developed an elaborate recognition system that fully exploits word shape properties to derive a ranked neighborhood of dictionary words that may match the unknown. First, a few "global" word features are used to filter the lexicon, eliminating unlikely candidates. A multitude of word-level features are then used to describe the distribution of strokes, edges, endpoints, loops, ascenders and descenders, etc. Since no segmentation is attempted, these features describe different regions of the word, rather than the individual characters. The recognition is then performed by combining the results from 41 classifiers, each using different attributes of the word image.

2.4 Discussion

The previous sections gave a summary of some of the main approaches to the problem of text recognition. This section will discuss some issues involved in those techniques that deal with the recognition of machine-printed text.

As illustrated by the short summary in section 2.2, significant work has been done on the problem of isolated character recognition (ICR). There are however inherent problems with using ICR as the basis for text recognition, most of which are because of the variable nature of the printed text itself. Some of these text attributes are outlined below.

Character Classes:

Since many ICR methods rely on some kind of feature extraction and comparison between individual unknown characters and the character prototypes, such systems must be *trained* extensively so that a good internal representation of each class of characters can be formed. This can be a time- and resource-consuming process.

Fonts:

Another problem is that ICR is sensitive to font variations. Therefore, such systems will probably have to be trained on a given font before they can recognize documents printed in that font. This would mean, to make a robust system, it would have to be trained on at least hundreds of different, common fonts and styles. Even after such training, the system may still have difficulties with “new” fonts.

Image Quality:

Perhaps the biggest drawback to using ICR as the single recognition technique is its high sensitivity to character (image) degradation. This is because most ICR methods, if used independently, would rely only on the structural attributes of the character image. The presence of broken characters, touching characters, streaks or specks can degrade the performance of such systems.

Many attempts have been made to overcome these problems. Some OCRs solve the problem by only expecting a small set of fonts and sizes of characters, on which they have had prior training. Also there are systems that accept font *descriptions* before a document in the given font is to be recognized. These approaches, however, limit robustness, and may only be suitable for limited applications [1].

A better approach is to use contextual knowledge in postprocessing of the results from ICR. Some of these methods were briefly touched upon in section 2.3.1. The effective use of context is important in all pattern recognition problems [35], and can lead to marked improvement in OCR performance. However, with OCRs, it usually entails the maintenance of a large dictionary.

A quite different and unique method has been proposed by Casey and Nagy [6]. They present a character recognition technique that does not rely on prior knowledge of the physical attributes of the characters (e.g. font, size) in a given document, and hence requires no training. It is also more tolerant to degradation in the character images than most ICR techniques. In their approach, Casey and Nagy exploit the available English letter-pair (bigram) frequencies, which show the probability of the occurrence of a given letter pair. Table 2.1 shows selected bigram probabilities for the English alphabet [31]. These values, which can be readily calculated using a large volume of online text, remain relatively constant for most English documents.

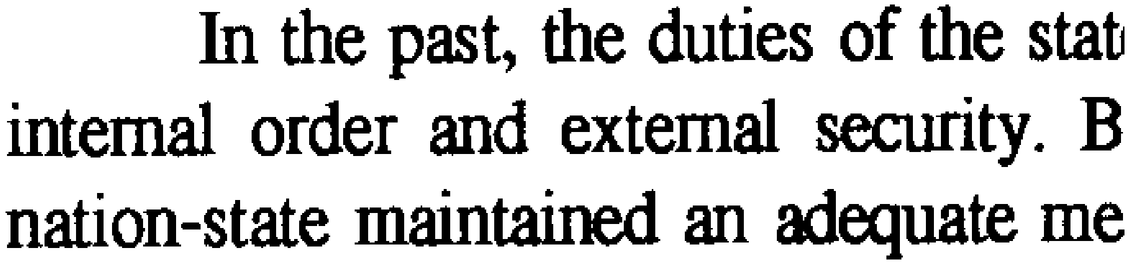
Letter	a	b	...	r	s	...	z
a	0.000	0.154	...	0.819	0.795	...	0.659
...	...	...	...	...	...	...	...
n	0.200	0.00	...	0.005	0.314	...	0.001
o	0.050	0.071	...	0.993	0.214	...	0.002
...	...	...	...	...	...	...	...
z	0.10	0.00	...	0.00	0.00	...	0.005

Table 2.1: Bigram frequencies for selected letter pairs

The system first groups "similar" characters into many separate *clusters*, which are refined until 26 groups of unknown characters are formed. It then calculates the bigram frequencies for these groups, capturing the *cluster*-pair relationships. These values are repeatedly "matched" against the *a priori* bigram frequencies, until a best match is achieved. By correctly matching the two tables, each of the clusters of unknown characters can be assigned the letter label of the corresponding class.

It is interesting to note that even such a robust method, which does not rely on the structural features of the characters, is not completely immune to image degradation and font variations. For example, problems of misclassification were reported when certain "difficult" fonts were used.

Another problem with character-based or segmentation-based systems is their basic need for correct segmentation of words into characters. As a consequence, these techniques can only work well if the scanned text is easily segmentable, since



In the past, the duties of the stat
internal order and external security. B
nation-state maintained an adequate me

Figure 2.8: Many instances of touching characters are present in scanned text

a general-purpose segmentation algorithm that does not require recognition does not exist. However, by observing most printed text sources, e.g. books, articles, etc., one can see that, in general, many touching characters are present. This is evident in figure 2.8, a scanned image of an actual text sample which was produced by a common typesetter.

Holistic word recognition methods (described in section 2.3.3) seem to represent a good alternative to methods that rely solely on character recognition. These techniques are particularly attractive because they do not require character segmentation, and are more tolerant to image degradation than ICR methods.

Another reason holistic word recognition presents an attractive alternative, is the apparent resemblance of the technique to how humans read text. Hull points out the fact that according to psychological experiments, humans do not read text character-by-character, rather they recognize *words* or even small *groups* of nearby words while they are reading:

> Interesting parallels are apparent when human reading process is compared to developments in algorithms for reading digital images of text. For example, the observation that fluent reading is not done on a character-by-character basis suggests that to achieve human competence in an algorithm, some word-level processing should take place before characters are identified. Also, an explicit identification of individual characters may not be an essential part of a reading algorithm. (*Hull* [19])

Although there are many advantages to holistic approaches to word recognition, they do not currently represent a complete solution. For example, they require large font-specific training sets and do not take advantage of language characteristics the way other character-based techniques have.

Chapter 3

Proposed Solution

A word recognition technique is proposed that avoids many problems associated with the traditional text recognition methods. Taking into account the importance of alternative knowledge sources, this method uses the structural similarity between equivalent words, as well as information about language characteristics such as word redundancy and inter-word relations.

Segmented word images are first passed through a matching step to determine word equivalency. A clustering algorithm is used to derive groups of equivalent word images without actual recognition of the words. A second step then uses the available language information such as *word frequencies* and *word transition probabilities* to generate hypotheses for the identity of the word contained in each group.

The steps involved in this method are illustrated in figure 3.1. The following sections give a more detailed description of the two main processes.

3.1 Word Image Matching by Clustering

3.1.1 Introduction

In many applications, it is often desirable to analyze the data in terms of groups of similar entities. A *clustering* algorithm is one that partitions a collection of data into distinct sets, each consisting of "similar" data points. In a *hierarchical*

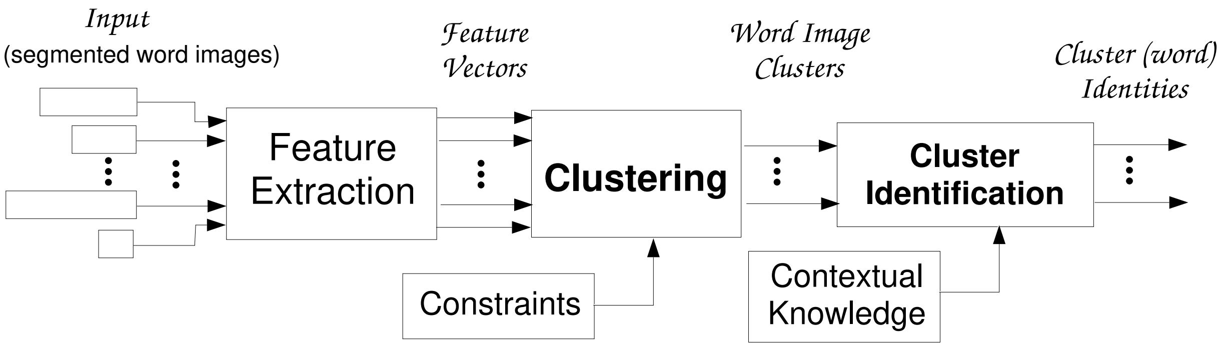


Figure 3.1: The main components of the recognition technique

clustering scheme, each cluster is itself classified into distinct groups [12]. Figure 3.2 illustrates a binary hierarchical cluster of some geometrical shapes, where every similar pair of shapes, or "clusters" of shapes, are joined repeatedly. Each sub-cluster represents another level of similarity, as the resulting tree is traversed.

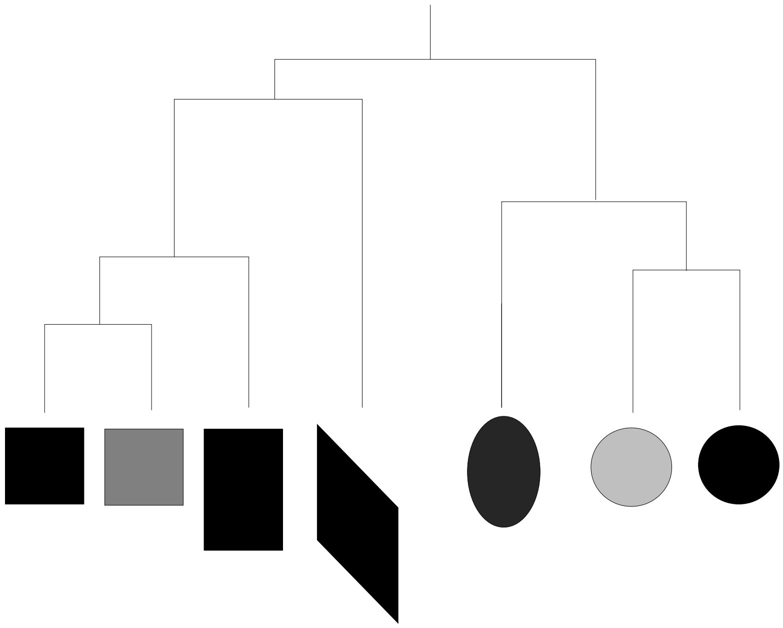


Figure 3.2: A hierarchical cluster of some geometrical shapes

Since the word recognition method proposed here relies on redundancy information, it would be useful to classify words as separate groups, each containing all the occurrences of a single word in the text passage. To do this, a modified hierarchical clustering algorithm is used. This process would, in the ideal case, produce a single cluster for each unique word in the passage. In many clustering methods, the number of different classes to be located must be known *apriori*. This is not necessary, however, for a hierarchical clustering algorithm, making the technique suitable for word matching, where the number of unique words in the passage is not known.

As illustrated by the example in figure 3.2, this particular clustering scheme operates by repeatedly merging similar data points or similar clusters. A description of the clusters must be derived in order to determine the similarity among them. Since the recognition technique deals with clusters of word images, by observing the structural features of all the words contained in each cluster, we can produce a description for it. Once a description for the cluster is obtained, we can compute the similarity (or difference) measure between each pair of clusters. By minimizing this "distance function", the most similar pair of clusters can be located, and can then be "merged" to produce a new cluster (hierarchy). The new set of clusters is then searched for the next closest pair, which are again merged, and so on.

3.1.2 The Role of Constraints

The reader may have already observed a possible drawback to the method described above, if it is applied to word matching. A simple hierarchical clustering algorithm will continue the merging process until it arrives at one main cluster containing a hierarchy of all the previously merged clusters (as in figure 3.2). To be effective for word matching, however, the clustering process must somehow be regulated, so that the number of resulting clusters is comparable with the (unknown) number of unique words in the test sample. For this purpose, a set of "constraints" are used.

Each constraint is a condition involving the merge candidates – the cluster pair that are currently most similar. If the cluster pair can satisfy all the constraints, they are merged to form a new cluster. Otherwise, the merge is avoided, and the next closest pair is located. If at some point in the process no more merges are possible due to these constraints, the clustering procedure is terminated, resulting in the remaining set of distinct groups. Since the constraints are to be used to regulate the merging process, they must reflect the likelihood of equivalency between the two candidate clusters. That is, the constraints must be designed in such a way as to *accurately* reflect inequality, if two clusters can not satisfy one or more of them. Otherwise, there is a risk of over-regulating the merge process.

The following is a list of the constraints incorporated into the clustering algorithm which is used in the word matching methodology. The description for the "strict" and "loose" characterization of the constraints is also given below.

1. Strict Constraints

By strict, it is meant that these constraints do not change across samples: they rely on facts and no degree of flexibility is allowed in their use.

Word non-adjacency Language statistics [22] reveal that it is highly improbable for two adjacent words in an edited passage of running text to have the same identity. This is especially true for frequent function words. For example, one rarely encounters a sentence such **as as** this one in a normal document[1]. Table 3.1 shows the low probabilities of such cases[2].

word pair	prob.	word pair	prob.
had had	$1e^{-2}$	that that	$1e^{-3}$
in in	$1e^{-4}$	a a	$9e^{-5}$
and and	$7e^{-5}$	is is	$1e^{-4}$
it it	$1e^{-4}$	of of	$3e^{-5}$
the the	$1e^{-5}$	with with	$1e^{-4}$

Table 3.1: Equivalent adjacent words are highly improbable

The cluster merge constraint described here uses this powerful *language* constraint to decide whether the merge candidates are in fact clusters containing two different words. If the first cluster contains a word that is adjacent to a word in

[1] Except in legal cases such as "*had had*" and "*that that*"

[2] Calculated from a text collection of over one Million words

the second cluster, then the two words, and therefore the two clusters, cannot have the same identity, and the merge is preempted. Note that the positions of the word images are known, since they are segmented in the order that they appear in the text.

2. Loose Constraints

Loose constraints refer to those conditions that may depend on some attributes of the text being analyzed. For example, a measure of degradation in the text can be used to adjust the flexibility of such constraints.

Word Image Length The length of an image refers to the width of its bounding box, measured in pixels. When dealing with text printed in the same font and size, the length of the images for equivalent words should be *approximately* the same. Therefore, two clusters are only merged if the average length of their word images are similar.

Word Image Density The density of an image reflects the percentage of black pixels in it. By the same reasoning as above, the merge of clusters that contain words with widely different densities is avoided.

Cluster Distance As indicated in the previous section, the distance between the candidate pair is calculated before their merge. If the resulting distance is too large, then it is unlikely that the two clusters contain equivalent word images.

Inter-level Distance Increase An indication of cluster inequality can also be obtained by observing the rate of increase in the distance measure calculated before each merge. For example, merging the two clusters shown in figure 3.3 ("so" and "an") will result in an unusually high rate of increase in the distance measure corresponding to the two levels: the top of the "so" cluster, and the top of what would be the "so-an" cluster. The algorithm is able to detect such cases, and preempt the merge.

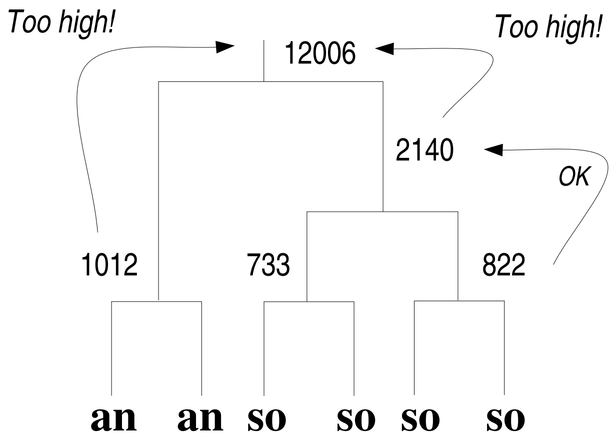


Figure 3.3: "Inter-level Distance Increase" constraint

3.2 Cluster Identification

The focus of the first stage of this recognition approach is to locate and group equivalent word images based on visual similarity and contextual knowledge. The second stage involves the actual *recognition* via cluster identification, whereby each cluster is assigned a word "tag", or a set of possible words (neighborhood). The following sections will illustrate how this can be achieved without actual recognition of the individual word images.

3.2.1 General Approach

A statistical study of the English language reveals that often there are unique relations among certain words or classes of words. Some of these are based in the grammar of the language, where grammatical rules impose certain restrictions on the words in a given sentence. These syntactical constraints can help in recognition of text, or disambiguation of neighborhoods [20]. Restricted patterns can also be found in the relationship between individual words. These are particularly interesting when the words appear frequently, resulting in more stable constraints.

Table 3.2 shows the top 20 most frequent words in English, and their frequencies. Note that these words account for almost 30 percent of any passage of text. Also, as it will be shown later, such words possess some unique relations. It is proposed that such information can be used in recognition of words, given their context, i.e. the given passage of text.

word	**freq.**	**word**	**freq.**	**word**	**freq.**	**word**	**freq.**
THE	0.071	IN	0.022	ON	0.008	AS	0.006
OF	0.032	FOR	0.010	HE	0.007	BY	0.006
AND	0.024	THAT	0.009	AT	0.007	IT	0.005
A	0.024	IS	0.008	WITH	0.006	HIS	0.005
TO	0.023	WAS	0.008	BE	0.006	SAID	0.004

Table 3.2: Top 20 most frequent words in English

Word Transition Probabilities

One set of probabilities that can describe the relationship among these words are the *word transition probabilities* (WTP). These reflect the likelihood of adjacency for a particular word pair (w_1w_2). Specifically

$$P(w_2 \mid w_1) = \frac{|w_1w_2|}{|w_1|}$$

where $\mid w_1w_2 \mid$ and $\mid w_1 \mid$ refer to the number of occurrences of the pair w_1w_2 and of the word w_1, respectively. This will show the ratio of all the w_1's which are followed immediately by the word w_2, reflecting the frequency of the pair.

Since it is impossible to gather statistics from every printed document, one can calculate good estimates by observing a very large body of running text. One such source is the Brown Corpus, a collection of more than one Million words of running text, whose composition will be explained in more detail in section 5.1. Assuming the Brown Corpus represents written, edited American English, the WTPs for the top 10 words in the language were calculated, and the results are shown in table 3.3. The rows and columns represent the first and second word in the pair, respectively. Thus, for example, the WTPs for the pair "*for the*" can be located by observing the value at the intersection of row 7 with column 1. Note that the blank cells in the table represent zero values, or ones that are lower than 0.0005.

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.283			0.040				0.004		
and	0.079	0.005		0.027	0.009	0.010	0.003	0.009	0.006	0.005
a										
to	0.130			0.026				0.002		
in	0.173		0.001	0.039			0.001	0.002		
for	0.223			0.088		0.002		0.003		
that	0.147	0.018		0.025	0.003	0.013	0.002	0.002	0.027	0.006
is	0.080	0.004		0.102	0.027	0.017	0.003	0.023		
was	0.068	0.001		0.067	0.017	0.024	0.003	0.006		

Table 3.3: Transition probabilities for top 10 words in English

Almost instantly, one can observe some apparent patterns in the relations among the above set of words. For example, the transition for the pair "*of the*" is stronger than any other – more than 28% of all the "*of* "s are followed by a "*the*". Also, note that "a" is not followed by any of the other top 10 words, while "and" follows an insignificant number of other words.

Graph Representation

The relations among a finite set of words can be visualized as a weighted connected digraph (directed graph). Thus, each word will represent a vertex, and the weights on the directed edges will reflect the degree of relatedness among the words. These weights can be the same as, or a function of, the WTPs for any given pair. Figure 3.4 is a (partial) graph representation for table 3.3 above.

Using Word Transition Probabilities

The clustering process results in clusters of equivalent, but yet unknown words. However, visual as well as contextual information about the words in each cluster are known, including the position of each word in the text. This will allow for the calculation of the WTP statistics for each *cluster* in the sample by examining the relations of the *words* contained in them. That is, for each cluster pair (i,j), $1 \leq i \neq j \leq N$, where N is the number of (a subset of) the clusters, one can calculate the number of words in cluster i that are immediately proceded by a word in cluster j. This in turn can be used to calculate the WTP for the two clusters using the equation on page 28. Using these statistics, along with other contextual and visual information about the words, one can attempt to match the resulting graph for the clusters to the corresponding graph for the apriori statistics. In this process, it is important to take into account not only the relation between a particular pair of words, but also each word's relations with the other words in the sample.

3.2.2 Matching Techniques

To understand the nature of the matching problem, it will be useful to start with an illustrative example. The overall stages of the clustering and matching processes are illustrated in figure 3.6. Segmented word images (figure 3.6(a)) are clustered, and a subset of clusters, corresponding to frequent function words, is chosen (figure 3.6(b)). As evident by the left graph in figure 3.6(c), corresponding to the inter-*cluster* transitions, this only represents a portion of all the possible transitions for the set of words. This is to be expected, as some less-frequent word transitions may not occur in the current sample. Nevertheless, there is sufficient information for an attempted match to a subset of the *apriori* transition graph. By matching the vertices in the observed graph to some of the vertices in the apriori graph, taking into account the weighted edges, the word associated with each vertex can be identified.

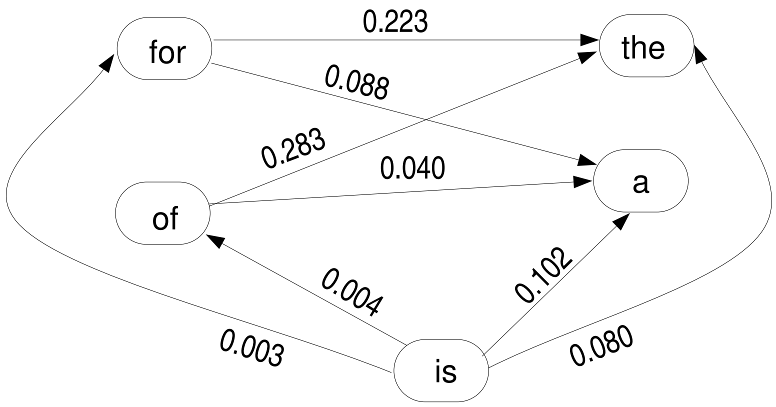


Figure 3.4: A transition graph for some of the top 10 words

The problem of matching the apriori and the observed graphs resembles some of the more general problems in graph theory. Particularly, a study of the graph *isomorphism* and *matching* problems is required. A brief description of the graph matching problem follows, but for a more complete treatment of this and other graph theory concepts, the reader is referred to [25], [26], or other textbooks on the subject.

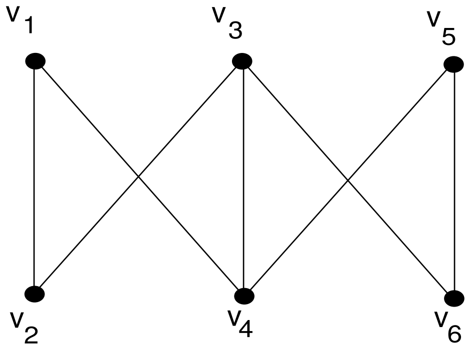


Figure 3.5: A *bipartite* graph with $V_1 = \{v_1, v_3, v_5\}$ and $V_2 = \{v_2, v_4, v_6\}$

Graph Matching

A graph $G = (V, E)$ is called *bipartite* if the vertex set $V = V_1 \cup V_2$, where V_1 and V_2 are disjoint, and further, for each edge $(v_1, v_2) \in E$, $v_1 \in V_1$ and $v_2 \in V_2$. Figure 3.5 shows an example of a bipartite graph.

The *matching problem* deals with detecting the correlation between two disjoint sets of vertices in a graph, such as V_1 and V_2 above. In other words, a set of graph edges must be located such that no two members of the set are adjacent. Furthermore, one can obtain a *maximum-cardinality matching* containing the most number of edges. For example, a maximum matching for the graph in figure 3.5 includes all the edges in the graph.

Application of Graph Matching

The goal is to find the correlation among the vertices in the *apriori* graph $G_a = (V_a, E_a)$, and observed graph $G_o = (V_o, E_o)$. That is, each vertex $v_{o_i} \in V_o$ must be "matched" to a vertex $v_{a_j} \in V_a$. This is somewhat similar to finding the maximum matching between the two sets V_a and V_o.

There are other issues involved, however. It is not sufficient to simply pair-up vertices from the two sets. Rather, in determining the matching, one must take into account whether each vertex pair has similar characteristics. These characteristics can be visual, such as a feature vector, or contextual, such as the way each word (vertex) relates to the others in the corresponding graph. This way, a degree of cross-graph similarity can be obtained for each vertex pair (v_{a_i}, v_{o_i}). This is referred to as the *weight* corresponding to the edges resulted by each matching. By maximizing these cross-graph weights, one can attempt to find the *maximum weight* matching.

3.3 Discussion

This section will explain some of the advantages of the recognition approach outlined above. The main drawback to isolated character recognition techniques is their high sensitivity to image quality. As explained in section 2.4, such methods have difficulties in recognition of text which contains many instances of touching or broken characters, or other kinds of image degradation. Going beyond character-level details, the word-matching algorithm determines equivalency by examining the shape of the *word* images. This makes the technique more tolerant to the usual kinds of noise mentioned above. Furthermore, by clustering equivalent word images, the algorithm, in effect, is compensating for small imperfections in each isolated instance of a word. That is, the recognition results can be improved by considering a collective description of the word, rather than isolated ones.

When dealing with printed text from sources such as newspapers, journals, and legal documents, it is usually the case that most of the text from the same source is printed in the same font. There is a good chance, however, that the font used would vary across different documents. Since most ICR methods must know the shape of each letter in advance, using such systems on documents from different sources may entail costly and time-consuming training of the system on each of the different fonts it may encounter. In addition, there may always be fonts that the system has not been trained on, and hence can not recognize. In contrast, since the word matching algorithm focuses on *similarity* among words in a given passage of text, it is not sensitive to font-changes across documents. Such a system would be able to determine that two words printed in the same font are equivalent, not because it is familiar with the font, but solely on the basis of visual similarity.

Perhaps a more broad benefit of the overall approach is that it embodies some high-level concepts associated with human reading. The collective information obtained from the visual shape of the *words*, redundancy in language, and the relationship among words provides a broad knowledge-base for any recognition system, only a fraction of which has been studied here.

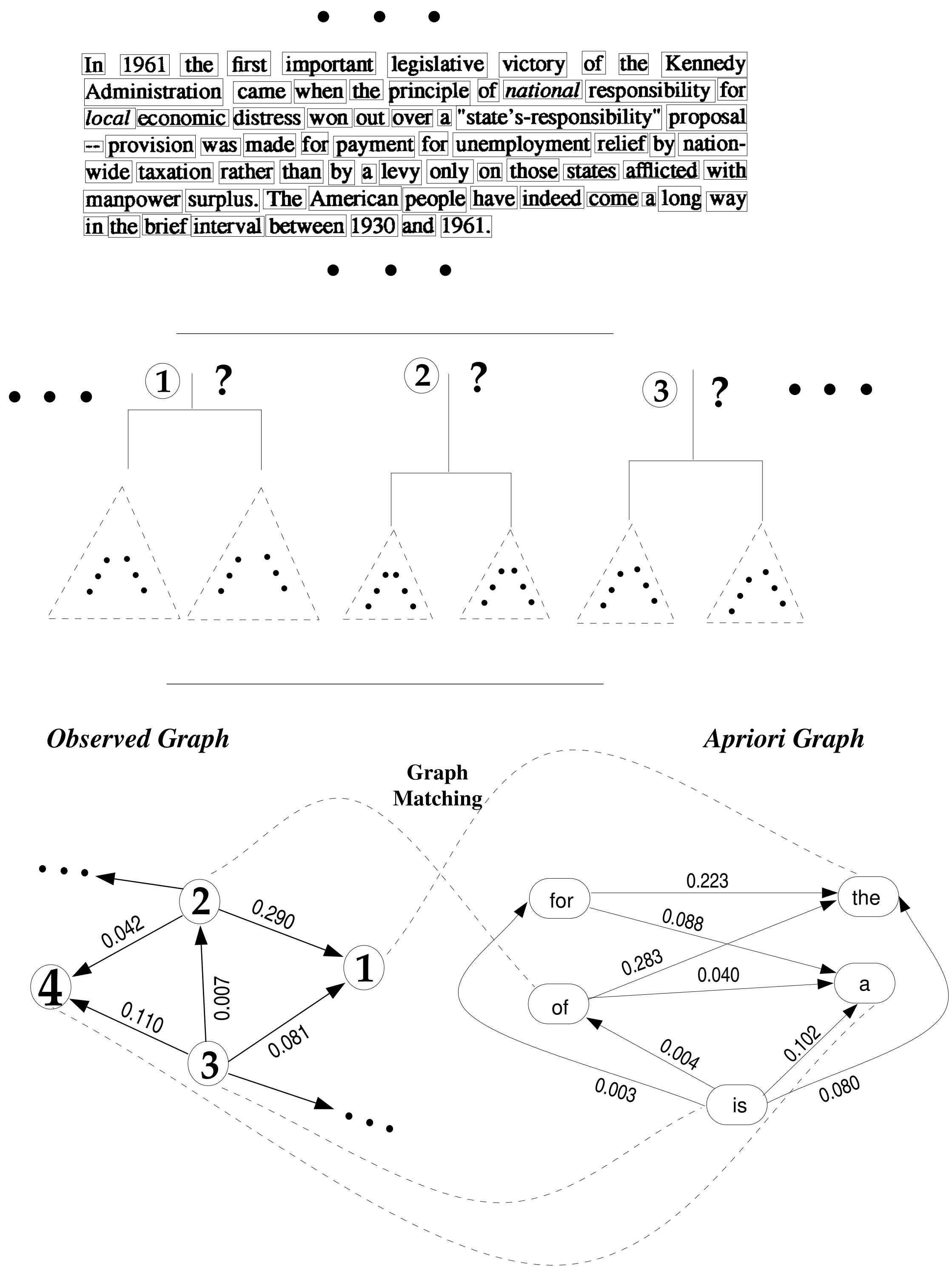


Figure 3.6: Cluster Identification by Graph Matching

Chapter 4

Details of the Solution

The two components of the proposed word recognition method are *clustering* and *cluster identification*. The algorithmic details of each part of the technique are presented in this chapter.

4.1 Clustering

The algorithm shown below is the top-level structure of the clustering component of the system. As reflected by the hierarchical nature of the algorithm, the task of the top-level loop is to find the next closest cluster pair among the clusters in the set, merge them, and replace the two with the new group. The loop is terminated when the two clusters that happen to be most similar are also constrained, meaning that no more merges of clusters should be allowed. Since no direct check for constraint satisfaction is performed at this level, the current distance between the pair is checked against infinity, which is by definition the distance between two constrained clusters.

{Initializations}
for each word images
    create a single cluster for image
    add the cluster **to** *cluster_list*
end for

Done ⟵ FALSE
while ¬*Done* **do**
    find_min_pair (*cluster_list, cluster1, cluster2, min_distance*)
    if *min_distance* $< \infty$ **then**
        temp ⟵ **merge** (*cluster1*,*cluster2*)
        remove *cluster1* and *cluster2* **from** *cluster_list*
        add *temp* **to** *cluster_list*
    else {*min_distance* is $\infty \Rightarrow$ no more merges are allowed}
        Done ⟵ TRUE
    end if
end while

The following sections discuss each of the main steps of the algorithm in more detail.

4.1.1 Cluster Initialization

A cluster may represent a single image or a pair clusters, each of which may in turn contain a pair of clusters, a pair of images, or a hybrid combination. Since initially no similarity measure is established among the word images, each word image is contained in a separate cluster.

Each cluster contains not only other clusters, but also some information about the current composition of itself. This helps the algorithm to check the status of the clustering process, and perform detailed analysis of each cluster. The data includes the total number of images contained in the cluster, the average dimensions for those images, and a feature vector describing the *centroid* of the current cluster.

For each of the initial clusters (containing only one word image) the centroid vector is equivalent to the image's feature vector. Principally, any type of feature vector can be used to describe the word images. However, the feature used must possess two important characteristics: it should be relatively intolerant to noise such as touching or broken characters, and it should describe the global (holistic) characteristics of the word image. Previous work has shown that the *stroke direction* *distribution* feature set is well-suited for word recognition problems[17]. Figure 4.1 shows an example of how the direction at each black pixel in an image is determined, and figure 4.2 shows the result for the strokes in a whole word image.

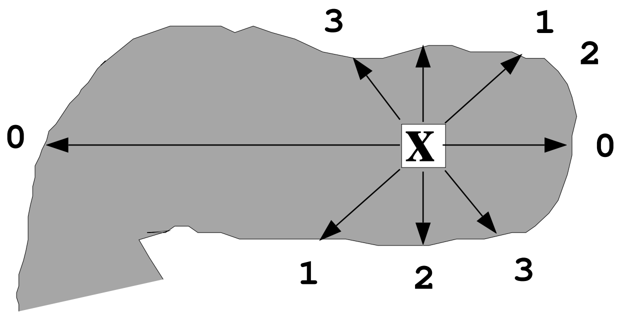


Figure 4.1: Pixel x has direction 0 (east-west)

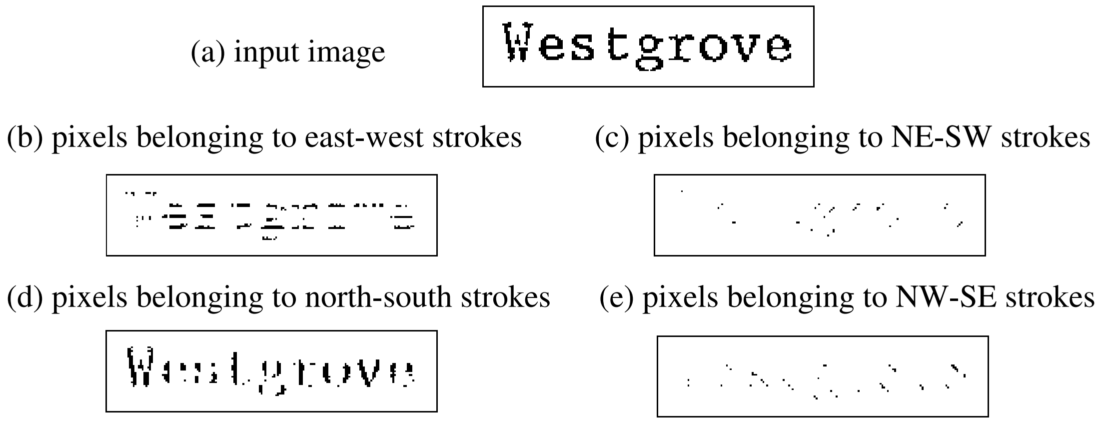


Figure 4.2: Stroke Direction Distribution

4.1.2 Finding the Closest Cluster Pair

The procedure **find_min_pair** in the main algorithm is responsible for locating, from the current set of clusters, the pair that is most similar. It then returns the two clusters, as well as the measure of their similarity, called their "distance". The distance which is returned, however, will be "infinity" if the closest pair have not been able to satisfy all the constraints, described in section 3.1.2.

Distance Calculation

A similarity measure between two clusters can be obtained by calculating a *distance* value using their feature descriptions. As previously stated, the single-image clusters are described by the feature vector for that image, while the centroid vector is used to describe multi-image clusters. By minimizing the *distance function* over the current set of clusters, the most similar pair can located. The distance function that is used is described by the following equation:

$$D_{hk} = \alpha_i d_{hi} + \alpha_j d_{hj} + \beta d_{ij} + \gamma \mid d_{hi} - d_{hj} \mid$$

where clusters i *and* j have already formed cluster k whose distance with cluster h is being measured, and **d** corresponds to the *squared Euclidean distance.* The above distance function is a *combinatorial* one, since the distance between the clusters can be obtained by using pre-existing measures for the sub-clusters, as illustrated in figure 4.3.

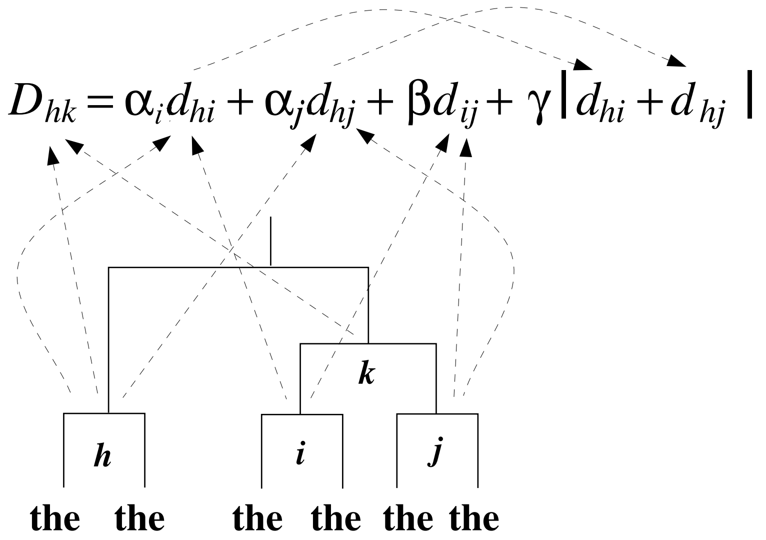


Figure 4.3: The distance function uses measures already computed

Method	α_i	α_j	β	γ
Nearest Neighbor	$\frac{1}{2}$	$\frac{1}{2}$	0	$-\frac{1}{2}$
Farthest Neighbor	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$
Median	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{4}$	0
Group Average	$\frac{N_i}{N_k}$	$\frac{N_j}{N_k}$	0	0
Centroid	$\frac{N_i}{N_k}$	$\frac{N_j}{N_k}$	$-(\alpha_i \cdot \alpha_j)$	0

Table 4.1: Parameters for various clustering schemes

Furthermore, it is possible to simulate any of a number of different classificatory algorithms by choosing appropriate values for the constants α_i, α_j, β, and γ [23]. Table 4.1 shows five clustering schemes that can be generated using the above distance function, and the corresponding parameters for each method.

The Use of Constraints

The constraints described in section 3.1.2 are used in conjunction with the distance measure to determine the feasibility of equivalence between two clusters. Here, the use of contextual information allows one to regulate the merging process. That is, if two clusters cannot satisfy all of the constraints, the algorithm can preempt their possible merge by declaring ∞ as the distance between them. Thus, they will not be chosen as the pair with the lowest distance, and **find_min_pair** will try to locate another pair. The top-level algorithm terminates when the least possible distance between cluster pairs <u>is</u> ∞, indicating that all remaining clusters are constrained. Algorithmic details of the distance calculation are shown below.

calculate_distance (*cluster1*, *cluster2*) <u>**returns**</u> <u>**real**</u>

{ N_1 : number of elements in cluster 1

N_{lk} : number of elements in the k^{th} subcluster of cluster l where $l, k \in \{1, 2\}$

$\vec{\imath}, \vec{\jmath}, \vec{h}$: feature vectors

$\alpha_i, \alpha_j, \beta, \gamma$ values pertain to the *Centroid* clustering strategy

function **SED** returns the squared Euclidean distance between two vectors}

$$\alpha_i \leftarrow \begin{cases} 1 & \text{if } \textit{cluster1} \text{ is a single image} \\ \frac{N_{11}}{N_1} & \text{otherwise} \end{cases}$$

$$\alpha_j \leftarrow \begin{cases} 1 & \text{if } \textit{cluster1} \text{ is a single image} \\ \frac{N_{12}}{N_1} & \text{otherwise} \end{cases}$$

$\beta \leftarrow -(\alpha_i \cdot \alpha_j)$

$\gamma \leftarrow 0$

$$\vec{\imath} \leftarrow \begin{cases} \textit{cluster2.}\vec{\textit{centroid}} & \text{if } \textit{cluster2} \text{ is a single image} \\ \textit{cluster2.subcluster1.}\vec{\textit{centroid}} & \text{otherwise} \end{cases}$$

$$\vec{\jmath} \leftarrow \begin{cases} \textit{cluster2.}\vec{\textit{centroid}} & \text{if } \textit{cluster2} \text{ is a single image} \\ \textit{cluster2.subcluster2.}\vec{\textit{centroid}} & \text{otherwise} \end{cases}$$

$\vec{h} \leftarrow \textit{cluster2.}\vec{\textit{centroid}}$

$\textit{distance} \leftarrow \alpha_i \cdot \mathbf{SED}(\vec{h}, \vec{\imath}) + \alpha_j \cdot \mathbf{SED}(\vec{h}, \vec{\jmath}) + \beta \cdot \mathbf{SED}(\vec{\imath}, \vec{\jmath}) + 0$

{check for *Constraints* before reporting the results}

if **CONSTRAINED** (*cluster1*, *cluster2* , *distance*) **then**

  distance $\leftarrow \infty$

end if

return *distance*

end calculate_distance

As shown above, if the two clusters are constrained, a distance of ∞ is returned by the function **calculate_distance**. If no other cluster pairs with a finite distance exist, then the procedure **find_min_pair** had to pass ∞ to the top-level loop, setting the termination condition.

4.1.3 Merging Clusters

The function **find_min_pair** using the above algorithm locates the next candidate pair, if there is one. The next step in main algorithm is to merge the two clusters, creating a new "parent" cluster for them, and returning the pertinent information. This includes computing new values for the total number of elements $(N_1 + N_2)$, average dimensions, and the *centroid* vector for the newly-formed cluster. Then the new cluster replaces both candidates in the set of current "live" clusters.

4.2 General Word Matching

This section deals with word matching by means of visual analysis *only*. The uses for such a method is explained, as well as an implementation.

4.2.1 Introduction

The clustering algorithm outlined above results in more refined clusters for shorter, more frequent words, as well as for isolated cases of very long words. There are cases, however, when the clustering scheme does not result in perfect clusters.

An alternative approach can be used to determine equivalency among "average" words whose clusters are usually shared with other such words. Also, such an approach can help to *refine* pre-existing clusters of frequent words, as a post-processing step for the clustering algorithm.

4.2.2 Word Autocorrelation

The technique described to perform word matching by visual means is similar to the one used in [3]. Two word images are "super-imposed" on one another, and the similarity or difference between them is calculated. Equivalency between a word and the top choices in its *neighborhood* can then be determined by observing the distance measures, and choosing the images that are sufficiently similar. The three main steps in the process are outlined below.

I. Baseline Estimation

The upper and lower baselines refer to the lines just below and just above the lower-case *a, o, c* and so on (see figure 2.7). These reference lines are very important to many recognition methods, and can reveal much information about the character or word image. The baselines are used here to align two word images correctly. That is, since the best *possible* match is sought, the baselines of the two images should be aligned when they are super-imposed, so that more accurate results can be obtained.

The baseline estimation is performed using a simple method based on the assumption that the distance D between the upper and lower reference lines is known. This is an assumption that should hold for all the words printed in the same font. The baselines are then the upper and lower boundaries of the band of height D which covers the most dense area of the image. This area will approximately represent the "middle" region of the word, and thus the baseline coordinates are determined.

Once the estimation is performed, any two word images can be aligned using their baselines. Also, to prepare images for the steps that follow, white regions are added as required to equalize the dimensions of the image pair.

This baseline estimation method is sufficient for the purpose of experimental study of word autocorrelation. Simply stated, if the two words are equivalent, their baselines should be consistent, even if the reference lines are not located perfectly. A more sophisticated estimation algorithm, such as the one used in [18], can be employed to improve accuracy.

II. Word Autocorrelation

By "super-imposition" of two binary images, it is meant that the pixels in one image are compared with the corresponding pixels in the other. This way, the pixel-by-pixel similarity or difference between the image pair can be observed, and the result can be used to form a new image. Simply, each pixel in coordinate [i,j] of the new image is set equal to the *exclusive-or* of the corresponding pixels in the two images. That is,

<u>for</u> each pixel coordinate $[i, j]$ **<u>do</u>**
    <u>if</u> *image1*$[i, j] \neq$ *image2*$[i, j]$ **<u>then</u>**
        output$[i, j] \leftarrow$ BLACK
    <u>else</u>
        output$[i, j] \leftarrow$ WHITE
    <u>end</u> if
<u>end</u> for

The resulting image then contains black regions where the two images differ.

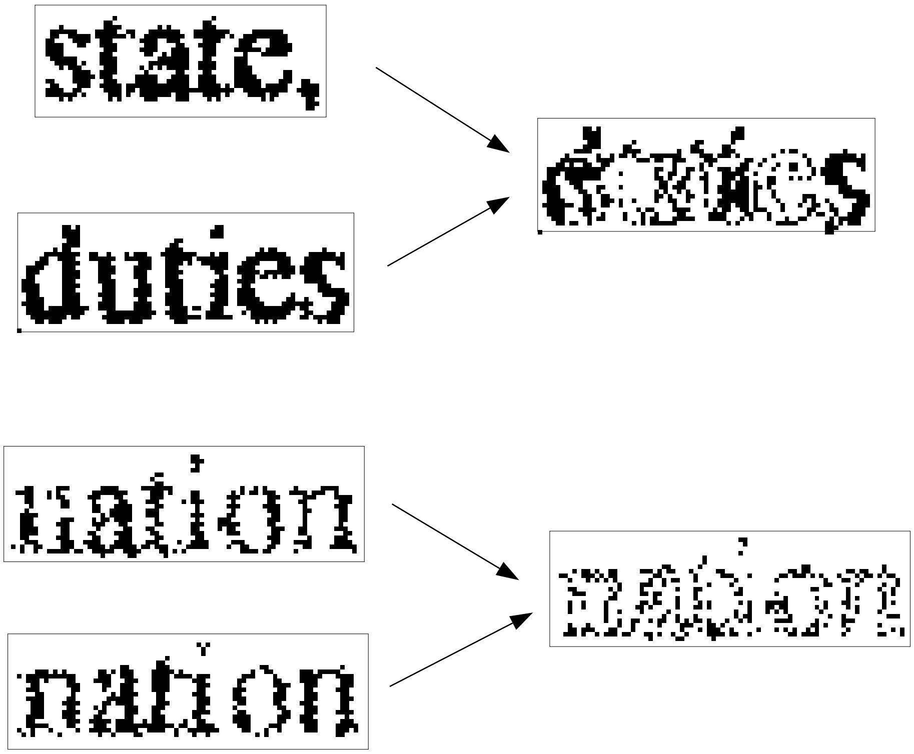


Figure 4.4: Word Autocorrelation results for sample word pairs

Just by visual examination or by counting the number of resulting black pixels, one should be able to get a good measure of similarity between the two images involved. However, as explained below, more formal methods may be needed.

III. **Distance Mapping**

Figure 4.4 shows two instances of the application of the word autocorrelation algorithm. The first involves two words with different identities, hence the autocorrelation process results in an image which has large black areas (blobs). The second example shows two words that are equivalent, but contain much noise. In each case, if the difference is determined by simply counting the number of black pixels, the resulting distances may be very close for both pairs of images. A robust algorithm, however, must use the fact that the "blobs" in the first case are much more significant than the disparate black regions in the second case.

Such an analysis can be performed by measuring the *distance map* [8] for the "correlated" image of each pair. Distance mapping is a method by which the distance of each black pixel to the nearest boundary of the given black region is calculated. Obviously, the interior pixels will have higher values than the ones close to a boundary. Using the distance map, one can distinguish between random noise, and meaningful, more dense black regions. A detailed algorithm to create a distance map for a binary image can be found in [8]. Figure 4.5 contains the numerical results corresponding to the distance map for a circle of radius 5.

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	1	1	1	2	1	1	1	0	0	0
0	0	1	1	2	2	2	2	2	1	1	0	0
0	1	1	2	2	3	3	3	2	2	1	1	0
0	1	1	2	3	4	4	4	3	2	2	1	0
0	1	2	3	4	5	6	5	4	3	2	1	0
0	1	2	2	3	4	4	4	3	2	2	1	0
0	1	1	2	2	3	3	3	2	2	1	1	0
0	0	1	1	2	2	2	2	2	1	1	0	0
0	0	0	1	1	1	2	1	1	1	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.5: Numerical distance map for a "circle" of radius 5 pixels

The distance map can also be displayed visually, as shown in figure 4.6. The darker, internal areas represent high values, while the lighter areas close to edges have lower values. The white background regions represent a "distance" value of zero.

A simple way for determining the difference between two images is to create the distance map for their "correlated" image, and then calculate the *summary* distance for the pair, using for instance, the sum-of-the-squares of all the values in the map. This will produce higher numbers for dense "blobs", such as the ones in the top example of figure 4.4, while resulting in low values for the less significant black regions

Figure 4.6: Visual representation of distance maps for different shapes

such as the ones in the second part of figure 4.4. This is evident in the experimental results even after the distance measure has been normalized by the average density of the original images.

Finally, figure 4.7 illustrates the steps involved in determining the distance between two words, for two separate cases. In each case, the images have been subjected to high levels of noise, as apparent by ragged and broken characters.

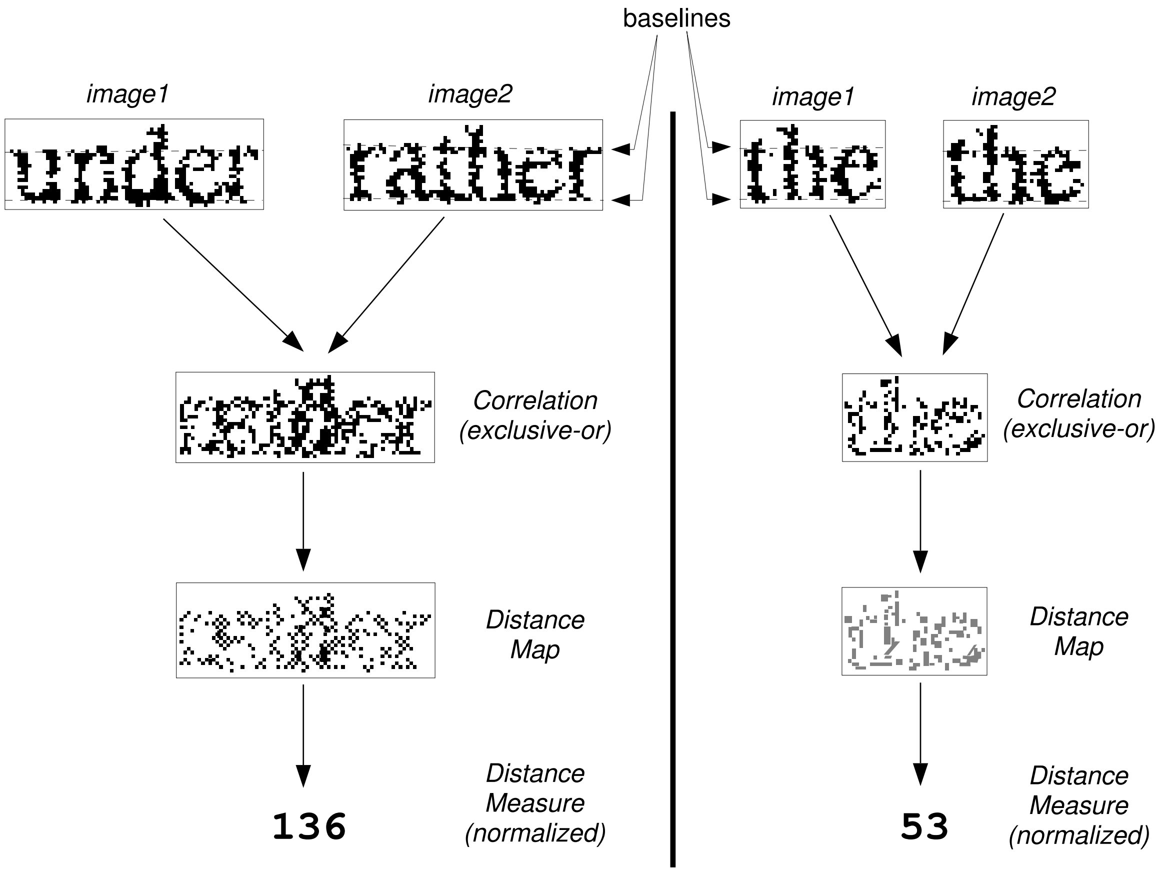


Figure 4.7: Word image comparison using the Word Autocorrelation method

Chapter 5

Empirical Study

The results from experimentation with the methods and algorithms discussed in the preceding chapters are presented here. A brief description of the data generation scheme is given, followed by the results generated by the clustering algorithm.

5.1 Experimental Environment

It is well-known that different styles of writing are used, depending on the subject matter and the author. Since the method proposed in this thesis uses linguistic information, experimentation should include sufficiently large passages of *running* text from many different sources and styles. Furthermore, the selection of the text used in any study should be close to "real-life" in both appearance and content.

5.1.1 The Brown Corpus

The Brown Corpus is an *on-line* collection of over one Million words of actual excerpts from printed material published in the year 1961. It was carefully designed by Kučera and Francis [22] at Brown University to reflect modern "edited American English." This resulted in the division of the corpus, proportionally, into 15 subject categories, ranging from Scientific to Press Reportage to Humor. The

code	category	samples	code	category	samples
A	Press: Reportage	44	J	Learned & Scientific	80
B	Press: Editorial	27	K	Fiction: General	29
C	Press: Reviews	17	L	Fiction: Mystery & Detective	24
D	Religion	17	M	Fiction: Science	6
E	Skills & Hobbies	36	N	Fiction: Adventure & Western	29
F	Popular Lore	48	P	Fiction: Romance & Love Story	29
G	Belles Letters	75	R	Humor	9
H	Miscellaneous	30			

Table 5.1: The 15 *genres* in Brown Corpus and the number of samples in each

resulting composition of the corpus is shown in table 5.1.

The name of each category in table 5.1 is followed by the number of "samples" it includes, where each sample is approximately 2000 words of running text from a single source. This particular distribution was chosen to be proportional to the amount of material that is available in each category [22]. Two different versions were produced for each sample in the corpus.

Untagged Version

This version of the corpus is designed to (closely) represent the appearance of the original text which each sample was taken from. It includes the words, as well as embedded "graphic markers" which indicate the existence of typographical features such as quotations, apostrophes, and italics, or other features of the text such as paragraphs, headlines, etc. This was done mainly due to the state of technology at the time of creation, and the lack of a better (online) typesetting mechanism. Also, a location code is appended to the end of each line indicating the sample code, and the original line number. Table 5.2 contains a partial list of the graphic markers, a complete list of which can be found in appendix A.1.

Tagged Version

Since the Brown Corpus was originally designed for computational study of the language, another version of the corpus was produced in the 1070s in which

marker	meaning	marker	meaning
**A	’ (apostrophe)	**C	:
*I	?	**K	%
*Q	“ (begin quotation)	**=	begin bold face
*U	” (end quotation)	**$	end bold face
*N	begin major heading	*/	begin Roman numeral
*P	end major heading	*,	end Roman Numeral

Table 5.2: Some of the graphic markers used in the *untagged* corpus

tag	description	examples
.	sentence closer	. ; ? !
AT	article	a, the, no
JJ	adjective	small, political
NP	proper noun	Einstein, Africa
PP$	possessive personal pronoun	my, our
RB	adverb	now, apparently
VB	verb, base form	make, carry
VBD	verb, past tense	said, opened

Table 5.3: Some of the syntactic tags used in the *tagged* corpus

every word is tagged by its syntactical category. In addition to the syntactic tag, a sample/location tag is associated with each word, making it easier to locate. A partial list of the syntactic tags is shown in table 5.3, while a complete list is presented in appendix A.2.

5.1.2 Automatic Text Image Generation

The Brown Corpus can be used to generate the word images needed for text recognition experiments. However, some translation is needed to convert the on-line corpus to text that resembles the original sources. That is, the soft-copy files with embedded graphic codes should be converted to the corresponding formatted text.

The steps to produce word images from the Brown Corpus (BC) are as follows. First the untagged BC file is processed, changing every graphic code into the corresponding code for a text formatting package (e.g. *ditroff*). The formatted file can be printed resulting in a hard-copy which should be a close approximation of the given original. The printed page can then be scanned using a desktop scanner, and translated into an *image*. This, however, can be a tedious and time-consuming process. It would be desirable to eliminate the printing and scanning steps.

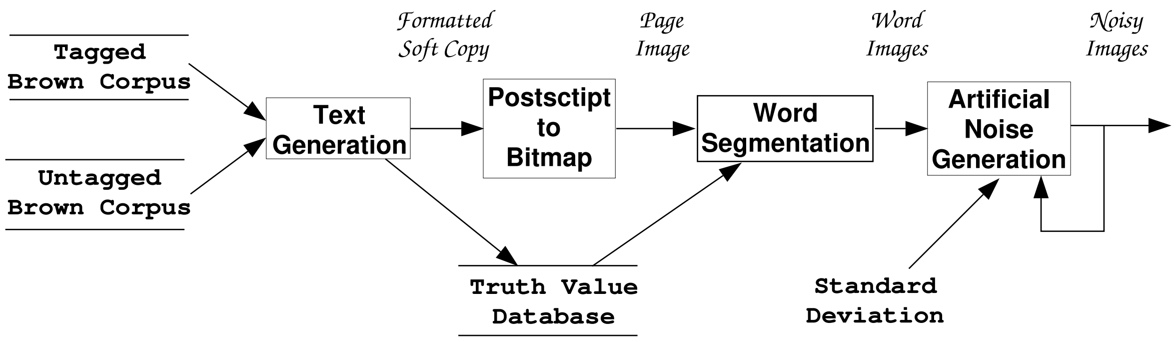


Figure 5.1: The automatic text generation process

Further processing can be done to generate a *bitmap* of the page, instead of printing it. This would allow for **automatic** generation of images from the soft-copy corpus, eliminating the need for printing every formatted page and scanning them. It is then possible to produce page images from any part of the BC almost instantly. As an example, figure 5.2 shows the original excerpt (from the original source) as well as its Brown Corpus untagged version, the soft-copy formatted (troff) file, and the final product, a formatted passage that resembles the original. Note the special graphical codes in the untagged version that help in retaining most of the typographical features.

Original

revolution. In 1961 the first important legislative victory of the Kennedy Administration came when the principle of *national* responsibility for *local* economic distress won out over a "state's-responsibility" proposal—provision was made for payment for unemployment relief by nation-wide taxation rather than by a levy only on those states afflicted with manpower surplus. The American people have indeed come a long way in the brief interval between 1930 and 1961.

Untagged BC

```
IN 1961 THE FIRST IMPORTANT LEGISLATIVE VICTORY OF THE *KENNEDY *ADMIN 0810E1G02
ISTRATION CAME WHEN THE PRINCIPLE OF *=NATIONAL*$ RESPONSIBILITY FOR * 0820E1G02
=LOCAL*$ ECONOMIC DISTRESS WON OUT OVER A **QSTATE**AS-RESPONSIBILITY* 0830E1G02
*U PROPOSAL**- PROVISION WAS MADE FOR PAYMENT FOR UNEMPLOYMENT RELIEF  0840E1G02
BY  NATION-WIDE TAXATION RATHER THAN BY A LEVY ONLY ON THOSE STATES AF 0850E1G02
FLICTED WITH MANPOWER SURPLUS. *THE *AMERICAN PEOPLE HAVE INDEED COME  0860E1G02
A LONG WAY IN THE BRIEF INTERVAL BETWEEN 1930 AND 1961.               0870E1G02
```

Ditroff Output

```
.nh
.nr .pp 11
.sz 11
.po 1.25i
.ll 4.5in
.pp
In 1961 the first important legislative victory of the Kennedy
Administration came when the principle of \fInational\fR
responsibility for \fIlocal\fR economic distress won out over a
"state's-responsibility" proposal -- provision was made for payment
for unemployment relief by nation-wide taxation rather than by a levy
only on those states afflicted with manpower surplus. The American
people have indeed come a long way in the brief interval between 1930
and 1961.
```

Generated Text Image

In 1961 the first important legislative victory of the Kennedy Administration came when the principle of *national* responsibility for *local* economic distress won out over a "state's-responsibility" proposal -- provision was made for payment for unemployment relief by nation-wide taxation rather than by a levy only on those states afflicted with manpower surplus. The American people have indeed come a long way in the brief interval between 1930 and 1961.

Figure 5.2: An example of the automatic text generation process

Word Segmentation

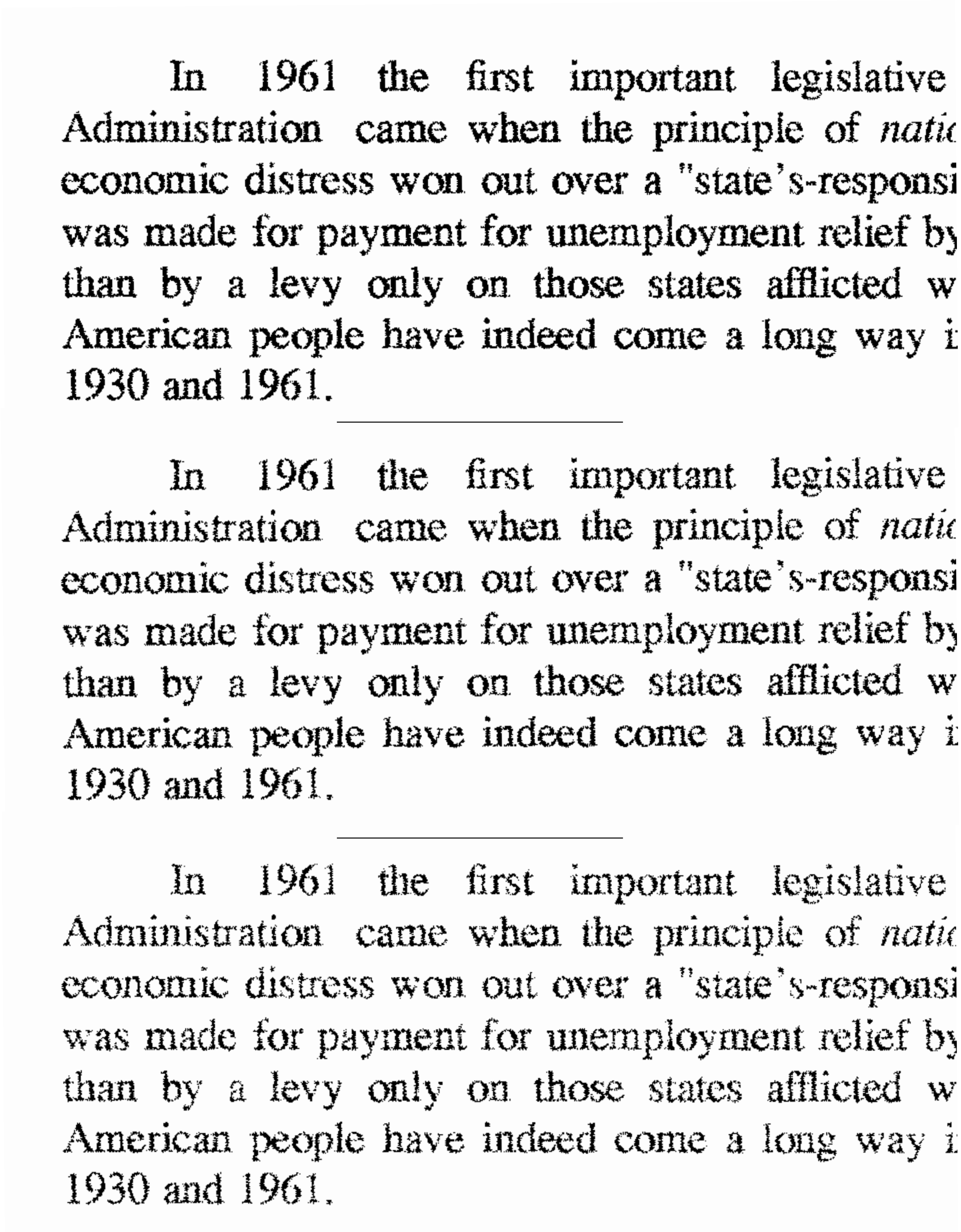
Once the page image is obtained, it is segmented into word images. Since the identity of the words are known as they are produced by the above text generation technique, each word image can be "truthed" as it is segmented. That is, the segmentation program uses a pre-produced soft-copy list of the words to attach the identity of each word to the corresponding image. This way, the results of clustering and recognition can be checked automatically.

The segmentation algorithm was developed using pre-existing techniques. First the "connected components" – regions of black pixels that are connected – are identified. Then, by assuming a minimum white space between words and lines, "close" connected components are joined together to form each word image, bounded by a one-pixel white boundary.

A useful feature incorporated into the word segmentation program is the automatic truth-value insertion. The image processing package used (HIPS [24]) allows for user-defined information to be added to an image's "header" – the image component preceding the actual bitmap. Taking advantage of this, the word segmentation program uses a soft-copy list of the words, and inserts the truth values in the corresponding image headers, as the words are segmented(see figure 5.4). This will give any other system which make future use of the images the ability to check its own results, and automatically calculate its error rates. The above steps will produce images that resemble words from the original source of the text, without manual reproduction. These images are, however, "clean", meaning they are in good condition, even if they contain many touching characters.

5.1.3 Introducing Noise

To experiment with degraded text, an Artificial Noise Generation process is applied to the word (or page) images to simulate some of the kinds of noise that are present in photocopies and facsimiles.



In 1961 the first important legislative
Administration came when the principle of *natic*
economic distress won out over a "state's-responsi
was made for payment for unemployment relief by
than by a levy only on those states afflicted w
American people have indeed come a long way i
1930 and 1961.

In 1961 the first important legislative
Administration came when the principle of *natic*
economic distress won out over a "state's-responsi
was made for payment for unemployment relief by
than by a levy only on those states afflicted w
American people have indeed come a long way i
1930 and 1961.

In 1961 the first important legislative
Administration came when the principle of *natic*
economic distress won out over a "state's-responsi
was made for payment for unemployment relief by
than by a levy only on those states afflicted w
American people have indeed come a long way i
1930 and 1961.

Figure 5.3: Three noise levels generated by two iterations with s.d.: 20,20 (top), 30,30(midle), 50,60(bottom)

In the noise model, summarized by the algorithm below, random black pixels on the letter boundaries are set to white. This generates ragged letter shapes and/or broken characters depending on the noise level. The amount of such noise can be controlled by the *standard deviation* used, and the number of times the process is repeated. A high value for the standard deviation means more boundary pixels will be chosen to be removed. By iterating the process of boundary degradation, one can generate even more degraded images, simulating broken characters. Since obtaining many copies of actual (noisy) facsimiles may be difficult, this method provides an acceptable, consistent alternative for producing degraded images, allowing one to perform more extensive experiments.

{ **R** : function returning a random number drawn from a normal distribution with
    the given mean and standard deviation, and a seed
  N : number of iterations
  s : seed for **R**
  T : threshold value }
$s \longleftarrow 1$
for $i \longleftarrow 1$ **to** N **do**
  for each Black *pixel* on a boundary **do**
    $x \longleftarrow \mathbf{R}_x(0, \sigma^2, s)$
    if $(x > \mathrm{T})$ **then** *pixel* $\longleftarrow$ White
  end for
  increment(s)
end for

By assigning appropriate values to N and σ, the noise level can be adjusted. Figure 5.3 shows the effect of three different noise settings on an actual text image.

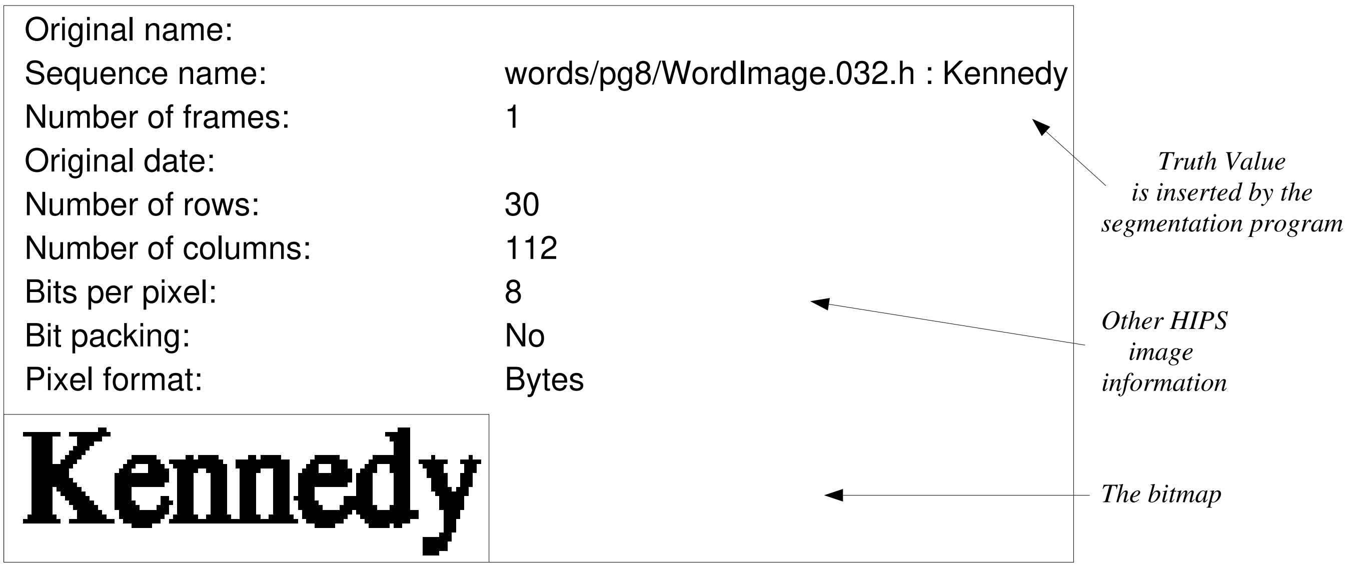


Figure 5.4: A sample word image and its header

5.2 Clustering Results

The results for a series of experiments with the clustering algorithm (sections 3.1, 4.1) are presented here. The effects of clustering *constraints*, as well as the effects of the sample size are investigated.

5.2.1 Error Analysis

Two general types of errors can be present in the clustering results: internal and external *fragmentation*. These kinds of mistakes are caused by excessive and insufficient clustering, respectively, as described below.

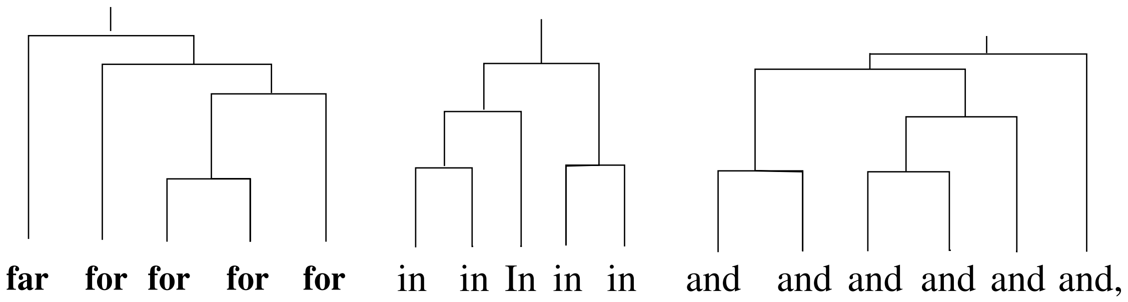


Figure 5.5: Three clusters with *internal fragmentation*

Internal Fragmentation

Internal fragmentation refers to the occurrence(s) of two or more *different* words in the same cluster. By different it is meant that either the words do not have the same identity, or if they do, some instances are capitalized, or have attached punctuation. Figure 5.5 shows several instances of such errors.

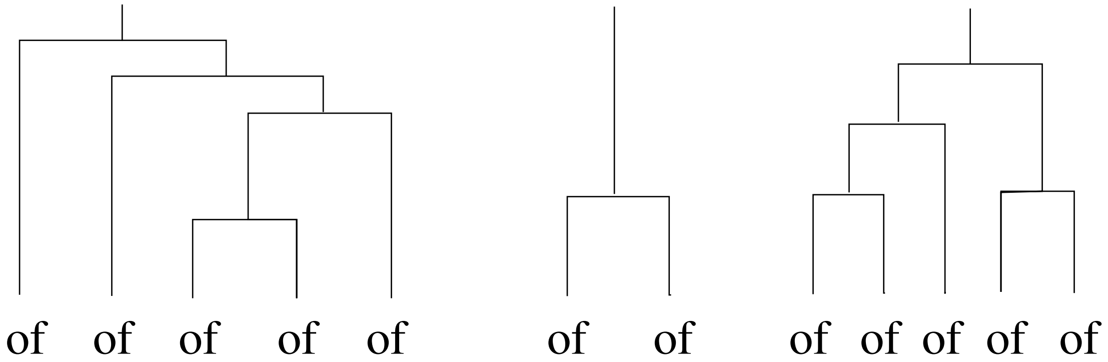


Figure 5.6: Examples of *externally* fragmented clusters

External Fragmentation

It is possible for some equivalent words to appear in separate clusters. This condition is referred to as external fragmentation, and an example of it is shown in figure 5.6.

5.2.2 Results

Experiments were performed using the clustering algorithm. Using the above text generation process, a sample of 2047 words of running text was produced corresponding to section G02 [1] of the Brown Corpus. Several experiments were performed on different-sized subsets of the sample, ranging from 50 to 2047 words.

Tables 5.4- 5.7 show the results for selected sample sizes. For each cluster corresponding to the top 10 frequent function words, the percentages for internal and external errors (fragmentations) are listed. The internal error represents the percentage of words in the cluster that are *not* equivalent to the base word. The external error is calculated by dividing the total number of words in the *smaller* cluster(s) by the total number of the base word in the sample. Note that only lower-case words are considered for these results, as the upper-case words, as well as those with attached punctuation, should be clustered separately. Also, if a word is not present in a particular sample, it is omitted from *average* measure calculation.

The best performance among in the above experiments was achieved with the sample of 1062 words, where there was no *external* fragmentation, and the average purity of the clusters for the top 10 words was **95.3**%. Generally, the performance tends to improve with larger samples. In the case of the test sample containing the entire G02 (2047 words), the performance decreased, but it is believed that it should improve with further refinement of the clustering algorithm.

To show the results with an entirely different passage of text, a sample (J10)[2] from category J (Learned and Scientific) of the Brown Corpus was chosen. The clustering results from this sample were also encouraging, and are shown in tables 5.11– 5.14. It is also interesting to observe the combined performance of the system on data sets from both samples (G02 and J10), as shown in table 5.15.

Experiments were also performed to study the effects of some of the *merge constraints* on the clustering performance. Tables 5.8 – 5.10 show the clustering

[1] from: Arthur S. Miller "Toward a Concept of National Responsibility", *The Yale Review*, December, 1961

[2] from: B.J.D. Meeuse, *The Story of Pollination*, New York: The Ronald Press Company, 1961. Pp. 104 - 108.

results for a sample of 1062 words, but each with some constraints missing. The numbering if the constraints are as follows:

1. Word Non-adjaceny
2. Image Length
3. Image Density
4. Cluster Distance
5. Cluster Distance Increase Rate.

These results confirm that the clusters play an important role in clustering, and a set of well-designed clusters can improve the performance dramatically.

Although the overall clustering results are promising in most cases, enhancements can be made to improve performance. By studying the results gathered, some of which is presented here, one can observe that most errors that do occur tend to be *internal* errors. That is, the algorithm tends to include words with different identities in the same cluster. One way to remedy this effect is further refinement of the merge constraints. A more effective approach would be to develop a better similarity measure, which can enable the algorithm to better distinguish among different words.

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	16	16	00.0	16	00.0
of	15	15	00.0	15	00.0
and	6	10	40.0	6	00.0
a	3+1	3+1	00.0	4	25.0
to	3	3	00.0	3	00.0
in	6	6	00.0	6	00.0
for	1	1	00.0	1	00.0
that	2	10	80.0	2	00.0
is	3	3	00.0	3	00.0
was	3	3	00.0	3	00.0
AVG.			**12.0**		**02.5**

Table 5.4: Clustering results for a sample of 200 words from G02

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	30	30	00.0	30	00.0
of	21	21	00.0	21	00.0
and	9	12	25.0	9	00.0
a	5	5	00.0	5	00.0
to	11	22	50.0	11	00.0
in	10	22	45.5	10	00.0
for	4	4	00.0	4	00.0
that	3	10	70.0	3	00.0
is	3	3	00.0	3	00.0
was	12	13	07.7	12	00.0
AVG.			**20.3**		**00.0**

Table 5.5: Clustering results for a sample of 400 words from G02

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	84	84	00.0	84	00.0
of	62	62	00.0	62	00.0
and	18	23	21.7	18	00.0
a	20	20	00.0	20	00.0
to	26	26	00.0	26	00.0
in	27	30	10.0	27	00.0
for	11	11	00.0	11	00.0
that	9	10	10.0	9	00.0
is	13	13	00.0	13	00.0
was	19	20	05.0	19	00.0
AVG.			**04.7**		**00.0**

Table 5.6: Clustering results for a sample of 1062 words from G02

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	167	167	00.0	167	00.0
of	114	114	00.0	114	00.0
and	45	48	08.2	45	00.0
a	22+13+1	22+13+1	00.0	36	38.9
to	53+1	58+1	8.6	54	01.9
in	45	45	00.0	45	00.0
for	16	24	33.3	16	00.0
that	25	37	33.4	25	00.0
is	39	39	00.0	39	00.0
was	21	32	34.4	2119	00.0
AVG.			**11.7**		**04.1**

Table 5.7: Clustering results for all of G02 (2047 words)

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	84	98	14.3	84	00.0
of	62	62	00.0	62	00.0
and	18	27	33.3	18	00.0
a	20	20	00.0	20	00.0
to	26	29	10.3	26	00.0
in	27	27	00.0	27	00.0
for	11	98	88.8	11	00.0
that	9	19	52.6	9	00.0
is	13	13	00.0	13	00.0
was	19	27	29.6	19	00.0
AVG.			**22.9**		**00.0**

Table 5.8: Clustering results for a sample of 1062 words from G02 without constraint 1

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	84	87	03.6	84	00.0
of	62	99	37.4	62	00.0
and	18	74	75.7	18	00.0
a	20	38	47.4	20	00.0
to	26	56	53.6	26	00.0
in	27	56	51.8	27	00.0
for	11	99	88.9	11	00.0
that	9	73	87.7	9	00.0
is	13	74	82.4	13	00.0
was	19	74	74.3	19	00.0
AVG.			**60.3**		**00.0**

Table 5.9: Clustering results for a sample of 1062 words from G02 without constraints 2 and 3

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	84	87	03.6	84	00.0
of	62	91	31.9	62	00.0
and	18	27	33.3	18	00.0
a	20	38	47.4	20	00.0
to	26	33	21.2	26	00.0
in	27	91	70.3	27	00.0
for	11	16	31.3	11	00.0
that	9	19	52.6	9	00.0
is	13	13	00.0	13	00.0
was	19	28	32.1	19	00.0
AVG.			**29.2**		**00.0**

Table 5.10: Clustering results for a sample of 1062 words from G02 without constraints 4 and 5

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	10	10	00.0	10	00.0
of	7	7	00.0	7	00.0
and	5	6	16.7	6	00.0
a	3	3	00.0	3	00.0
to	6	6	00.0	6	00.0
in	4+1+1	4+1+1	00.0	6	33.3
for	3	3	00.0	3	00.0
that	3	3	00.0	3	00.0
is	4+1	4+1	00.0	5	20.0
was	–	0	–	–	–
AVG.			**01.9**		**05.9**

Table 5.11: Clustering results for a sample of 200 words from J10

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	20	20	00.0	27	00.0
of	17	17	00.0	17	00.0
and	12	15	20.0	12	00.0
a	8+1	8+1	00.0	9	11.1
to	9	9	00.0	9	00.0
in	7+1	7+1	00.0	8	12.5
for	4	4	00.0	4	00.0
that	7	7	00.0	7	00.0
is	10	10	00.0	10	00.0
was	–	0	–	–	–
AVG.			**02.2**		**02.6**

Table 5.12: Clustering results for a sample of 400 words from J10

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	64	64	00.0	64	00.0
of	33	33	00.0	33	00.0
and	28+1	33+1	17.8	29	03.4
a	31	31	00.0	31	00.0
to	22	47	53.2	22	00.0
in	25	47	46.8	25	00.0
for	10	10	00.0	10	00.0
that	15	15	00.0	15	00.0
is	20+1	20+1	00.0	21	04.8
was	1	1	00.0	1	00.0
AVG.			**11.8**		**00.8**

Table 5.13: Clustering results for a sample of 1062 words from J10

word	no. in cluster	cluster total	% error internal	no. in sample	% error external
the	135	135	00.0	135	00.0
of	77+1	77+1	00.0	78	01.3
and	46+1	48+1	09.6	47	02.1
a	57	57	00.0	57	00.0
to	43	100	57.0	43	00.0
in	55	100	45.0	55	00.0
for	14	14	00.0	14	00.0
that	32	42	21.0	32	00.0
is	44	44	00.0	44	00.0
was	3	3	00.0	00.0	00.0
AVG.			**13.3**		**00.3**

Table 5.14: Clustering results for the entire J10 (2005 words)

sample	avg. % error	
size	internal	external
200	6.95	4.2
400	11.25	1.3
1062	8.25	0.4
entire	12.5	2.2

Table 5.15: Overall clustering results for both G02 and J10

5.2.3 Alternative Similarity Measure

As explained in section 4.2, other word matching methods, such as word autocorrelation can be used to refine the results of the clustering process. To explore the feasibility of this method, and to determine the degree to which it can aid clustering, some small experiments were performed.

Figure 5.7: Three noise levels used in Word Autocorrelation experiments.

The experiments summarized below were performed using a sample of 500 word images reproduced from section G02 of the Brown Corpus. Three separate experiments were conducted using the "clean" images, as well as 2 different sets of noisy images, with medium (s.d. 30 & 30) and high (s.d. 50 & 60) levels of degradation. The noise was introduced into the images using the method outlined in section 5.1.3. Sample words with the three degradation levels are shown in figure 5.7.

	total in	no noise		medium noise level		high noise level	
word	sample	no. error	% jump	no. error	% jump	no. error	% jump
the	34	0	143	0	30	0	19
of	25	0	16	2	12	2	2
was	12	0	80	0	31	0	26
world	4	0	148	0	32	0	12
territorial	3	0	301	0	101	0	80

Table 5.16: Examples of specific word matching results in a 500 word sample

Table 5.16 shows the autocorrelation results for selected words, exemplifying the general results. The three columns correspond to the three different levels of noise. The error in each case corresponds to the number of words which had a *different* identity, but were found before the last occurrence of the base word in the neighborhood. Also, each "% jump" column refers to the percent increase in the distance values between the last occurrence of the base word, and the word immediately following it in the neighborhood. This may be helpful in locating the "threshold" – the location of the last equivalent word to the base – in each neighborhood. That is, by scanning the distance increases in the neighborhood, one may be able to locate the last occurrence of the base word.

A restriction imposed on the above experiments with word autocorrelation involved word length. Obviously, words with a large differential in their word length will have high distances. Therefore, only words with similar lengths (within 10 pixels) were compared using the word autocorrelation method.

5.3 Cluster Identification

There are some important language characteristics that can be used to help in cluster identification. One such statistic is word frequency. Comparing the *apriori* word frequencies with the observed *cluster populations*, may aid in the identification of the clusters. Table 5.17 shows such statistics for the sample of 1062 words from G02. For each word, the expected frequency (defined by the language) is followed by the actual frequency in the sample and the corresponding frequency exhibited by the *cluster* population.

word	**predicted freq.**	**actual freq.**	**cluster freq.**
the	0.071	0.079	0.079
of	0.032	0.058	0.058
and	0.024	0.017	0.022
a	0.024	0.019	0.019
to	0.023	0.025	0.024
in	0.022	0.025	0.028
for	0.010	0.010	0.010
that	0.009	0.009	0.009
is	0.008	0.012	0.012
was	0.008	0.018	0.019

Table 5.17: Various frequencies all for a sample of 1062 words from G02

To study the feasibility of using the word transition probabilities for cluster identification, some statistics were extracted from the Brown Corpus. A comparative study was performed to determine the effect, if any, different categories of writing have on the transition patterns. In addition to individual sections, the statistics were also calculated for groups of similar genres, such as the ones pertaining to press, fictional writing, and so on. Tables 5.18 shows the WTPs for sections A, B and C combined, reflecting the results for the "press" section of the corpus.

Table 5.19 show the results for all sections of the corpus, except "fiction" (K,L,M,N,P) and "humor" (R) sections (750000), while table 5.20 shows the results for the whole Brown Corpus. The fiction genres seem to exhibit erratic statistics, since they contain a more informal writing style, as well as much dialogue. Finally table 5.21 and 5.21 shows the WTPs for sections G and G02, respectively.

By inspection of these tables, we can detect certain patterns in word relations across different populations. The same kinds of patterns are, to a certain degree present in table 5.23, where the transition values for the *clusters* are shown. Obviously these values tend to stabilize as the sample size increases.

An algorithm can be developed which takes into account word frequencies and word transition probabilities, and derives the cluster identities. This can be done using a graph matching framework, as explained in section 3.2.2, and incorporating some heuristic rules about the nature of the English language. The clusters for frequent function words can be located by observing the top distance value and the standard deviation of the clusters. The frequent function words generally possess lower values for both. Once these clusters are located, the corresponding values for the word frequencies and WTPs can be calculated and matched to the *apriori* statistics.

Some examples of the heuristic rules mentioned above are as follows: if two frequent function word clusters both possess very high word frequencies, as well as a high transition value, and if their macro features (such as word length) are consistent, then the clusters may be those of "*of*" and "*the*". Other analysis such as examining the *in-degree* and the *out-degree* of each cluster (graph vertex) can also be performed.

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.285			0.042				0.004		
and	0.075	0.005		0.026	0.009	0.009	0.003	0.010	0.006	0.005
a										
to	0.136			0.029				0.003		
in	0.281		0.002	0.065	0.001		0.001	0.002		
for	0.235		0.001	0.097		0.001		0.004		
that	0.163	0.017		0.026	0.002	0.013	0.002	0.002	0.027	0.006
is	0.075	0.004	0.001	0.109	0.026	0.020	0.004	0.028		
was	0.067	0.001	0.004	0.069	0.019	0.024	0.003	0.007		

Table 5.18: Transition probabilities for top 10 words in samples A, B and C, combined (about 175,000 words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.283			0.040				0.004		
and	0.079	0.005		0.027	0.009	0.010	0.003	0.009	0.006	0.005
a										
to	0.130			0.027				0.003		
in	0.173		0.001	0.040	0.001	0.005	0.001	0.002		
for	0.223		0.001	0.088		0.002		0.003		
that	0.147	0.018		0.025	0.003	0.013	0.002	0.002	0.027	0.006
is	0.080	0.004	0.001	0.102	0.027	0.017	0.003	0.023		
was	0.068	0.001	0.004	0.067	0.017	0.024	0.003	0.006		

Table 5.19: Transition probabilities for top 10 words in samples A, B, C, D, E, F, G, H, and J combined (about 750,000 words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.283			0.040				0.004		
and	0.079	0.005		0.027	0.009	0.010	0.003	0.009	0.006	0.005
a										
to	0.130			0.026				0.002		
in	0.173		0.001	0.039			0.001	0.002		
for	0.223			0.088		0.002		0.003		
that	0.147	0.018		0.025	0.003	0.013	0.002	0.002	0.027	0.006
is	0.080	0.004		0.102	0.027	0.017	0.003	0.023		
was	0.068	0.001		0.067	0.017	0.024	0.003	0.006		

Table 5.20: Transition probabilities for top 10 words in all samples of BC (about 1 Million words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.256		0.001	0.039				0.006		
and	0.082	0.004		0.021	0.013	0.019	0.004	0.012	0.003	0.004
a										
to	0.131		0.001	0.026		0.001		0.005		
in	0.269		0.003	0.061	0.001		0.001	0.005		
for	0.197		0.002	0.082		0.001		0.008	0.001	
that	0.132	0.020	0.001	0.023	0.003	0.018	0.003	0.001	0.031	0.009
is	0.085	0.003		0.094	0.033	0.015	0.004	0.035		
was	0.063	0.004		0.092	0.035	0.020	0.003	0.012		

Table 5.21: Transition probabilities for top 10 words for section G (about 150,000 words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.386			0.009				0.009		
and	0.067				0.044	0.022		0.022		
a										
to	0.148									
in	0.244			0.044				0.022		
for	0.313									
that	0.192	0.039							0.039	0.039
is	0.154			0.103		0.026		0.026		
was	0.095			0.048	0.095					

Table 5.22: Transition probabilities for top 10 words for sample G02 (2047 words)

word	the	of	and	a	to	in	for	that	is	was
the								0.012		0.012
of	0.387			0.016						
and	0.043				0.043					
a										
to	0.192		0.077							
in	0.200							0.033		
for	0.455									
that	0.100	0.100								0.100
is	0.077			0.231		0.077				
was	0.050			0.050	0.100					

Table 5.23: Cluster transition probabilities corresponding to the top 10 words in a sample of 1062 words from G02

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.277			0.041				0.002		
and	0.099	0.006		0.021	0.012	0.014	0.004	0.013	0.007	0.004
a										
to	0.143		0.001	0.037				0.005	0.001	
in	0.294			0.065			0.001	0.006		
for	0.214			0.052		0.003		0.001		
that	0.213	0.026		0.023	0.005	0.019	0.005	0.001	0.033	0.007
is	0.083	0.009		0.072	0.022	0.013	0.002	0.015		
was	0.035			0.048	0.024	0.008	0.001	0.005		

Table 5.24: Transition probabilities for top 10 words for section J (about 190,000 words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.299			0.065						
and	0.152	0.022				0.022		0.022		
a										
to	0.093			0.093				0.023		
in	0.346			0.055			0.018			
for	0.143			0.071						
that	0.156	0.063		0.031		0.066			0.031	
is				0.044	0.044			0.022	0.022	
was										

Table 5.25: Transition probabilities for top 10 words for sample J10 (2005 words)

word	the	of	and	a	to	in	for	that	is	was
the										
of	0.242			0.091						
and	0.031	0.031			0.031	0.031		0.031		
a										
to	0.213			0.064			0.021			
in	0.213			0.064			0.021			
for	0.200			0.100						
that	0.200				0.133	0.133				
is				0.050	0.050	0.050				
was										

Table 5.26: Cluster transition probabilities for the corresponding to the top 10 words in a sample of 1062 words from J10

Chapter 6

Conclusions and Future Research

Even today, many OCR systems have difficulties with recognition of words with touching or broken characters. Some methods have used non-structural attributes of the letters, such as bigrams, but are still limited by the quality of the images. Other methods avoid such problems by recognizing whole words. However, this usually requires maintenance of a large lexicon, and extensive training with many different fonts.

A technique has been proposed that can solve the problems associated with traditional systems. Using structural (visual) equivalency among words, combined with contextual knowledge, clusters of equivalent words are constructed. It was shown that by using constraints, derived from contextual knowledge, the clustering results improve dramatically.

The clustering step of the approach is tolerant to touching characters, and image degradation such as broken characters, since it uses visual equivalency among words of the same identity. Also, since the object is producing groups of equivalent word images, no explicit training of the algorithm is required.

Results from several experiments with the clustering algorithm were presented. The method achieved up to 95.3% cluster accuracy for a sample of over 1,000 words. It was proposed that further improvements are possible by employing a more accurate distance measure.

For cluster identification, it was shown that *word transition probabilities* provide valuable information, and sometimes, well-defined patterns in inter-word relations. Using these patterns, and the ones acquired from the word clusters, one can establish a match, and locate the appropriate word identity for each cluster. One way to perform the match is by using a modified *graph matching* algorithm, whereby a graph edge connects each cluster with the corresponding word in the language. In case of ambiguity, the algorithm can derive a set of hypotheses for the cluster, forming a neighborhood. Other contextual information, such as *visual* relations among the words can be used to further refine the neighborhoods. Finally, a syntax-based analysis can help in locating the correct choice among the words in the neighborhood [20].

Some topics of interest for future study include new enhancements to the clustering algorithm. As it was demonstrated, the *word autocorrelation* method provides a good similarity measure even for noisy word images. This technique can be used in conjunction with the existing distance measure, leading to more pure clusters. Also, given the positive effect the *merge constraints* have on the performance, the design of more constraints should be considered.

One aspect of the clustering methodology that was not sufficiently studied is its ability to generate many different classificatory schemes. It would be interesting to observe the effects of using different clustering methods, as applied to word matching.

The experiments reported in this work have shown that more reliable statistics can be derived for larger text samples. To obtain a better measure of the performance of various algorithms, experiments with larger data sets are recommended. There is also a possibility of genre-dependence of some probabilities, as demonstrated by the "fiction" categories.

Bibliography

[1] Auerbach. *Auerbach on Optical Character Recognition.* Auerbach, 1971.

[2] Raimo Bakis, Noel M. Herbst, and George Nagy. An experimental study of machine recognition of hand-printed numerals. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(2):119–132, July 1968.

[3] Norman F. Brickman and Walter S. Rosenbaum. Word autocorrelation redundancy match (WARM) technology. *IBM Journal of Research and Development*, 26(6):681–686, November 1982.

[4] Evan L. Brill. Character recognition via fourier descriptors. In *WESCOM Technical Papers, Session 25*, 1968.

[5] Richard G. Casey and C. R. Jih. A processor-based OCR system. *IBM Journal of Research and Development*, 27(4):386–399, July 1983.

[6] Richard G. Casey and George Nagy. An autonomous reading machine. *IEEE Transactions on Computers*, C-17(4):492–503, May 1968.

[7] Richard G. Casey and George Nagy. Decision tree design using a probabilistic model. *IEEE Transactions on Information Theory*, IT-30(1):93–99, January 1984.

[8] Per-Erik Danielsson. Euclidian distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[9] A. C. Downton, E. Kabir, and D. Guillevic. Syntactic and contextual post-processing of handwritten addresses for optical character recognition. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 1072–1076, Rome, Italy, November 1988.

[10] R.A. Duderstadt, M.J. Cykana, A.V. Monfared, and D.H. Gibbs. Isolated word recognition for postal address processing. In *4th USPS Advanced Technology Conference*, pages 233–245, November 1990.

[11] L.D. Earnest. Machine recognition of cursive writing. In C.M. Popplewell, editor, *Information Processing 1962, Proceedings of the IFIP Congress 62*, pages 462–466, Munich, Germany, August 1962.

[12] Brian Everitt. *Cluster Analysis.* John Wiley and Sons, 1974.

[13] G. H. Granlund. Fourier preprocessing for hand print character recognition. *IEEE Transactions on Computers*, 21:195–201, February 1972.

[14] George Hannan. The RCA multi-font reading machine. In G. L. Fisher, Jr., D. K. Pollock, B. Raddack, and M. E. Stevens, editors, *Optical Character Recognition.* Spartan Books, 1962.

[15] Leon D. Harmon. Automatic recognition of print and script. *Proceedings of the IEEE*, 60:1165–1176, October 1972.

[16] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. A word shape analysis approach to recognition of degraded word images. In *4th USPS Advanced Technology Conference*, pages 217–231, Washington, D.C., November 1990.

[17] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Word recognition with multi-level contextual knowledge. In *Proceedings of the International Conference on Document Analysis and Recognition*, Saint Malo, France, September 1991.

[18] Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. A word shape analysis approach to lexicon based word recognition. *(to appear in) Pattern Recognition Letters*, 1992.

[19] Jonathan J. Hull. Hypothesis generation in a computational model for visual word recognition. *IEEE Expert*, 1(3):63–70, 1986.

[20] Jonathan J. Hull. Incorporation of a markov model of language syntax in a text recognition algorithm. In *Symposuim on Document Analysis and Information Retrievel*, Las Vegas, Nevada, March 16-18 1992.

[21] H. Kazmierczak. The potential field as an aid to character recognition. In *Proceedings of the International Conference on Information Processing, UNESCO*, pages 244–247, Paris, France, 1960.

[22] Henry Kučera and W. Nelson Francis. *Computational Analysis of Present-Day American English.* Brown University Press, Providence, Rhode Island, 1967.

[23] G. N. Lance and W. T. Williams. A general theory of classificatory strategies – 1. hierarchical systems. *Computer Journal*, 9(4):373–380, 1967.

[24] M.S. Landy, Y. Cohen, and G. Sperling. HIPS: A unix-based image processing system. *Computer Vision, Graphics, and Image Processing*, 25:331–347, 1984.

[25] Clifford W. Marshall. *Applied Graph Theory.* John Wiley and Sons, 1971.

[26] James A. McHugh. *Algorithmic Graph Theory.* Prentice Hall, 1990.

[27] Edward M. Riseman and Allen R. Hanson. A contextual postprocessing system for error correction using binary n-grams. *IEEE Transactions on Computers*, C-23:480–493, May 1974.

[28] A. A. Spanjersberg. Combination of different systems for the recognition of handwritten digits. In *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, pages 208–209, August 1974.

[29] R. J. Spinrad. Machine recognition of hand printing. *Information and Control*, 8:124–142, 1965.

[30] I. H. Sublette and J. Tults. Character recognition by digital feature detection. *RCA Review*, 23(1):60–79, March 1962.

[31] Ching Y. Suen. n-gram statistics for natural language understanding and text processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):164–172, April 1979.

[32] Ching Y. Suen, Marc Berthod, and Shunji Mori. Automatic recognition of hand-printed characters - the state of the art. *Proceedings of the IEEE*, 68(4):469–487, April 1980.

[33] L. A. Svyatogor. Algorithm for the recognition of typed symbols on the basis of ideal reference standards. In V. A. Kovalevsky, editor, *Character Readers and Pattern Recognition*. Spartan Books, 1968.

[34] J. T. Tou and R. C. Gonzalez. Recognition of handwritten characters by topological feature extraction and multilevel categorization. *IEEE Transactions on Computers*, C-21(7):776–785, July 1972.

[35] Godfried T. Toussaint. The use of context in pattern recognition. *Pattern Recognition*, 10:189–204, 1978.

[36] N. D. Tucker and F. C. Evans. A two-step strategy for character recognition using geometrical moments. In *Proceedings of the 2nd International Joint Conference on Pattern Recognition*, pages 223–225, August 1974.

[37] V. K. Yeliseyev. Character reader based on the principle of optical correlation. In V. A. Kovalevsky, editor, *Character Readers and Pattern Recognition*. Spartan Books, 1968.

Appendix A

Brown Corpus Markers

A.1 Graphics Markers

marker	meaning	marker	meaning
**A	' (apostrophe)	**U	" (end quotation)
*B	uncoded character(s)	**X	!
*C	:	**Y	lower-case Greek letter
*D	¨ (diaeresis or umlaut)	**Z	upper-case Greek letter
*F	formula	+	+ or &
*H	ellipsis	*(	[
*I	?	*)	]
*J	begin symbol	*$	end italic or underscoring
*K	%	*/	begin Roman numeral
*N	begin major heading	*,	end Roman numeral
*N**P	begin major division	*+0	deg (degree)
	without heading	**-	– (dash)
*P	end major heading	**=	begin bold face
*Q	" (begin quotation)	**$	end bold face
*R	begin minor heading	**(	begin capitalization
*R**T	begin minor division	**)	end capitalization
	without heading	**.	abbreviation period
*S	;	**..	abbreviation period
*T	end minor heading		at end of sentence

A.2 Syntactic Tags

tag	description	examples
.	sentence closer	. ; ? !
(	left paren	
)	right paren	
	not, n't	
–	dash	
,	comma	
:	colon	
ABL	pre-qualifier	*quite, rather*
ABN	pre-quantifier	*half, all*
ABX	pre-quantifier	*both*
AP	post-determiner	*many, next*
AT	article	*a, the, no*
BE	*be*	
BEDZ	*was*	
BEG	*begin*	
BEM	*am*	
BEN	*been*	
BER	*are, art*	
BEZ	*is*	
CC	coordinating conjunction	*and, or*
CD	cardinal numeral	*one, two, 2*
CS	subordinating conjunction	*if, although*
DO	*do*	
DOD	*did*	
DOZ	*does*	
DT	singular determiner	*this, that*
DTI	singular or plural qualifier	*some, any*
DTS	plural determiner	*these, those*
DTX	determiner/double conjunction	*either*
EX	existential *there*	
FW	foreign word	
HV	*have*	
HVD	*had* (past tense)	
HVG	*having*	
HVN	*had* (past participle)	
IN	preposition	
JJ	adjective	
JJR	comparative adjective	*chief, top*
JJS	semantically superlative adj.	*chief, top*
JJT	morphologic. superlative adj.	*biggest*

tag	description	examples
MD	modal auxiliary	*can, should*
NC	cited word (after reg. tag)	
NN	singular or mass noun	
NN$	possessive singular noun	
NNS	plural noun	
NNS$	possessive plural noun	
NP	proper noun	
NP$	possessive proper noun	
NPS$	possessive plural proper noun	
NR	adverbial noun	*home, today*
OD	ordinal numeral	*first, 2nd*
PN	nominal pronoun	*everybody*
PN$	possessive nominal pronoun	
PP$	possessive personal pronoun	*my, our*
PP$$	2nd possessive. pers. pronoun	*mine, ours*
PPL	sing. reflexive pers. pronoun	*myself*
PPLS	plur. reflexive pers. pronoun	*ourselves*
PPO	objective personal pronoun	*me, it, them*
PPS	3rd sing. nominative pronoun	*she, it, one*
PPSS	other nominative pers. pron.	*I, we, they, you*
QL	qualifier	*very, fairly*
QLP	post-qualifier	*enough, indeed*
RB	adverb	
RBR	comparative adverb	
RBT	superlative adverb	
RN	nominal adverb	*here, then*
RP	adverb/particle	*about, off, up*
TO	infinitive marker *to*	
UH	interjection, exclamation	
VB	verb, base form	
VBD	verb, past tense	
VBG	verb, present participle	
VBN	verb, past participle	
VBZ	verb, 3rd singular present	
WDT	*wh-* determiner	
WP$	possessive *wh-* determiner	
WPO	objective *wh-* determiner	
WPS	nominative *wh-* determiner	
WQL	*wh-* qualifier	
WRB	*wh-* adverb	

Appendix B

Clustering Output

The following is sample output from the clustering program showing the results for a sample of 100 words. The hierarchical composition of each cluster is shown in detail – levels of the hierarchy are distinguished by the indentation of each line. The output of each cluster is followed by a summary of the different sub-clusters found, including how many words of the same identity were in each sub-cluster, and what percentage of the overall cluster population it represents. Also, the standard deviation (S.D.) of each cluster is shown. The real number at the top of each multi-image cluster is the distance between its two subclusters, and the number to the left of each word is the index of the word in the passage. Note that the identity of the words are not determined by the clustering program, but are printed for readability.

```
==================
Cluster Number 000
==================

0 In

---
Found  1 sub-clusters:
In                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 001
==================

3322.7
```

```
S.D. =  118
(
 1359.0
 S.D. =    0
 (
  79 the
 )
 S.D. =   40
 (
   548.0
  S.D. =    0
  (
   70 the
  )
  S.D. =    0
  (
   35 the
  )
 )
)
S.D. =  146
(
 2686.1
 S.D. =    0
 (
  6 the
 )
 S.D. =  126
 (
  2543.2
  S.D. =   86
  (
    816.0
   S.D. =    0
   (
    66 the
   )
   S.D. =    0
   (
    31 the
   )
  )
  S.D. =  108
  (
   2398.0
   S.D. =    0
   (
    3 the
   )
   S.D. =    0
   (
    1 the
   )
  )
 )
```

```
)

---
Found  1 sub-clusters:
the                        8 ( 100.0 % )

Standard Deviation =    188

==================
Cluster Number 002
==================

2 past,

---
Found  1 sub-clusters:
past,                      1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 003
==================

98 body

---
Found  1 sub-clusters:
body                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 004
==================

4 duties

---
Found  1 sub-clusters:
duties                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 005
==================

2684.8
S.D. =  106
(
 3912.0
 S.D. =    0
 (
  69 of
```

```
 )
 S.D. =    0
 (
  60 of
 )
)
S.D. =  137
(
 2422.1
 S.D. =   94
 (
  2504.0
  S.D. =    0
  (
   99 of
  )
  S.D. =    0
  (
   33 of
  )
 )
 S.D. =  100
 (
  2268.9
  S.D. =    0
  (
   75 of
  )
  S.D. =   86
  (
   1800.5
   S.D. =    0
   (
    41 of
   )
   S.D. =   60
   (
    1458.0
    S.D. =    0
    (
     82 of
    )
    S.D. =    0
    (
     5 of
    )
   )
  )
 )
)

---
Found  1 sub-clusters:
of                           8 ( 100.0 % )
```

```
Standard Deviation =    173

==================
Cluster Number 006
==================

84 law"

---
Found  1 sub-clusters:
law"                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 007
==================

7 state,

---
Found  1 sub-clusters:
state,                      1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 008
==================

4388.8
S.D. =  181
(
 10230.0
 S.D. =    0
 (
  64 as
 )
 S.D. =    0
 (
  54 as
 )
)
S.D. =    0
(
 8 as
)

---
Found  1 sub-clusters:
as                          3 ( 100.0 % )

Standard Deviation =    229

==================
```

```
Cluster Number 009
==================

9 Sir

---
Found  1 sub-clusters:
Sir                          1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 010
==================

10 Henry

---
Found  1 sub-clusters:
Henry                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 011
==================

11 Maine

---
Found  1 sub-clusters:
Maine                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 012
==================

12 noted

---
Found  1 sub-clusters:
noted                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 013
==================

13 long

---
Found  1 sub-clusters:
```

```
long                        1 ( 100.0 % )

Standard Deviation =       0

==================
Cluster Number 014
==================

14 ago,

---
Found  1 sub-clusters:
ago,                        1 ( 100.0 % )

Standard Deviation =       0

==================
Cluster Number 015
==================

15 were

---
Found  1 sub-clusters:
were                        1 ( 100.0 % )

Standard Deviation =       0

==================
Cluster Number 016
==================

16 only

---
Found  1 sub-clusters:
only                        1 ( 100.0 % )

Standard Deviation =       0

==================
Cluster Number 017
==================

17 two

---
Found  1 sub-clusters:
two                         1 ( 100.0 % )

Standard Deviation =       0

==================
Cluster Number 018
==================
```

```
18 in

---
Found  1 sub-clusters:
in                           1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 019
==================

19 number:

---
Found  1 sub-clusters:
number:                      1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 020
==================

20 internal

---
Found  1 sub-clusters:
internal                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 021
==================

3397.0
S.D. =  134
(
 3510.0
 S.D. =    0
 (
  56 other
 )
 S.D. =    0
 (
  28 other
 )
)
S.D. =  125
(
 3066.0
 S.D. =    0
 (
```

```
  44 order
 )
 S.D. =    0
 (
  21 order
 )
)

---
Found  2 sub-clusters:
other                      2 ( 50.0 % )
order                      2 ( 50.0 % )

Standard Deviation =    183

==================
Cluster Number 022
==================

5478.0
S.D. =    0
(
 43 and
)
S.D. =    0
(
 22 and
)

---
Found  1 sub-clusters:
and                        2 ( 100.0 % )

Standard Deviation =    132

==================
Cluster Number 023
==================

23 external

---
Found  1 sub-clusters:
external                   1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 024
==================

9042.0
S.D. =  225
(
 9040.0
```

```
 S.D. =    0
 (
  62 equality,
 )
 S.D. =    0
 (
  42 certainty
 )
)
S.D. =    0
(
 24 security.
)

---
Found  3 sub-clusters:
equality,                  1 ( 33.3 % )
certainty                  1 ( 33.3 % )
security.                  1 ( 33.3 % )

Standard Deviation =    305

==================
Cluster Number 025
==================

25 By

---
Found  1 sub-clusters:
By                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 026
==================

26 prevailing

---
Found  1 sub-clusters:
prevailing                 1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 027
==================

27 over

---
Found  1 sub-clusters:
over                       1 ( 100.0 % )
```

```
Standard Deviation =      0

==================
Cluster Number 028
==================

80 sole

---
Found  1 sub-clusters:
sole                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 029
==================

29 claimants

---
Found  1 sub-clusters:
claimants                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 030
==================

30 for

---
Found  1 sub-clusters:
for                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 031
==================

85 (a

---
Found  1 sub-clusters:
(a                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 032
==================
```

```
32 loyalties

---
Found  1 sub-clusters:
loyalties                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 033
==================

86 term

---
Found  1 sub-clusters:
term                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 034
==================

34 men,

---
Found  1 sub-clusters:
men,                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 035
==================

94 coined

---
Found  1 sub-clusters:
coined                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 036
==================

36 nation-state

---
Found  1 sub-clusters:
nation-state               1 ( 100.0 % )

Standard Deviation =      0
```

```
==================
Cluster Number 037
==================

9036.0
S.D. =    0
(
 76 dominance.
)
S.D. =    0
(
 37 maintained
)

---
Found  2 sub-clusters:
dominance.                 1 ( 50.0 % )
maintained                 1 ( 50.0 % )

Standard Deviation =    184

==================
Cluster Number 038
==================

38 an

---
Found  1 sub-clusters:
an                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 039
==================

39 adequate

---
Found  1 sub-clusters:
adequate                   1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 040
==================

40 measure

---
Found  1 sub-clusters:
measure                    1 ( 100.0 % )
```

```
Standard Deviation =      0

==================
Cluster Number 041
==================

92 remember,

---
Found  1 sub-clusters:
remember,                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 042
==================

93 was

---
Found  1 sub-clusters:
was                        1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 043
==================

77 It

---
Found  1 sub-clusters:
It                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 044
==================

83 "international

---
Found  1 sub-clusters:
"international             1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 045
==================
```

```
45 within

---
Found  1 sub-clusters:
within                      1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 046
==================

46 its

---
Found  1 sub-clusters:
its                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 047
==================

47 territorial

---
Found  1 sub-clusters:
territorial                 1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 048
==================

48 borders.

---
Found  1 sub-clusters:
borders.                    1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 049
==================

49 Outside

---
Found  1 sub-clusters:
Outside                     1 ( 100.0 % )

Standard Deviation =      0
```

```
==================
Cluster Number 050
==================

50 those

---
Found  1 sub-clusters:
those                      1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 051
==================

51 limits

---
Found  1 sub-clusters:
limits                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 052
==================

 794.0
S.D. =    0
(
 88 it
)
S.D. =    0
(
 52 it
)

---
Found  1 sub-clusters:
it                         2 ( 100.0 % )

Standard Deviation =     28

==================
Cluster Number 053
==================

53 asserted,

---
Found  1 sub-clusters:
asserted,                  1 ( 100.0 % )
```

```
Standard Deviation =      0

==================
Cluster Number 054
==================

72 world,

---
Found  1 sub-clusters:
world,                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 055
==================

 942.0
S.D. =    0
(
 65 against
)
S.D. =    0
(
 55 against
)

---
Found  1 sub-clusters:
against                    2 ( 100.0 % )

Standard Deviation =     45

==================
Cluster Number 056
==================

81 "subject"

---
Found  1 sub-clusters:
"subject"                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 057
==================

57 states,

---
Found  1 sub-clusters:
states,                    1 ( 100.0 % )
```

```
Standard Deviation =      0

==================
Cluster Number 058
==================

1289.8
S.D. =   61
(
 2110.0
 S.D. =    0
 (
  73 a
 )
 S.D. =    0
 (
  97 a
 )
)
S.D. =    0
(
 58 a
)

---
Found  1 sub-clusters:
a                          3 ( 100.0 % )

Standard Deviation =     88

==================
Cluster Number 059
==================

1744.0
S.D. =    0
(
 74 position
)
S.D. =    0
(
 59 position
)

---
Found  1 sub-clusters:
position                   2 ( 100.0 % )

Standard Deviation =     79

==================
Cluster Number 060
==================
```

```
78 became

---
Found  1 sub-clusters:
became                     1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 061
==================

61 sovereign

---
Found  1 sub-clusters:
sovereign                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 062
==================

89 is

---
Found  1 sub-clusters:
is                         1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 063
==================

63 and, 33.62  19.70  18.43  32.42  29.45

---
Found  1 sub-clusters:
and,                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 064
==================

91 to

---
Found  1 sub-clusters:
to                         1 ( 100.0 % )

Standard Deviation =      0
```

```
==================
Cluster Number 065
==================

96 Bentham),

---
Found  1 sub-clusters:
Bentham),                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 066
==================

90 pertinent

---
Found  1 sub-clusters:
pertinent                  1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 067
==================

67 "inferior"

---
Found  1 sub-clusters:
"inferior"                 1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 068
==================

68 peoples

---
Found  1 sub-clusters:
peoples                    1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 069
==================

95 by
```

```
---
Found  1 sub-clusters:
by                           1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 070
==================

87 which,

---
Found  1 sub-clusters:
which,                       1 ( 100.0 % )

Standard Deviation =      0

==================
Cluster Number 071
==================

71 non-Western

---
Found  1 sub-clusters:
non-Western                  1 ( 100.0 % )

Standard Deviation =      0
```