Effects of OCR errors on text categorization

Lidija K Mackovski

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/1331

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
EFFECTS OF OCR ERRORS ON TEXT CATEGORIZATION

by

Lidija K. Mackovski

Bachelor of Science
Department of Electrical and Computer Engineering
University of St. Cyril and Methodius, Skopje, Macedonia
1996

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2001
The Thesis prepared by
Lidija K. Mackovski

Entitled
Effects of OCR errors on text categorization

is approved in partial fulfillment of the requirements for the degree of
Master of Science Degree - Department of Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Graduate College Faculty Representative

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ABSTRACT

Effects of OCR Errors on Text Categorizations

by

Lidija K. Mackovski

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In this thesis, we report on our experiments on training and categorization of optically recognized documents. In particular, we present a lexicon-based error correction algorithm to improve the categorization process. This algorithm is based on edit distance techniques and information from highly weighted words in the categorizers.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>v</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2 PRELIMINARIES</td>
<td>3</td>
</tr>
<tr>
<td>OCR</td>
<td>3</td>
</tr>
<tr>
<td>Text categorization</td>
<td>5</td>
</tr>
<tr>
<td>Dimensionality reduction</td>
<td>8</td>
</tr>
<tr>
<td>Methods for construction of a classifier</td>
<td>9</td>
</tr>
<tr>
<td>Effectiveness measures</td>
<td>12</td>
</tr>
<tr>
<td>CHAPTER 3 BACKGROUND</td>
<td>14</td>
</tr>
<tr>
<td>OCR errors</td>
<td>14</td>
</tr>
<tr>
<td>Tools for OCR error correction</td>
<td>17</td>
</tr>
<tr>
<td>Experimental environment</td>
<td>19</td>
</tr>
<tr>
<td>The experiments</td>
<td>21</td>
</tr>
<tr>
<td>CHAPTER 4 ERROR CORRECTION ALGORITHM</td>
<td>24</td>
</tr>
<tr>
<td>Design of the dictionary and the categorizers</td>
<td>25</td>
</tr>
<tr>
<td>The algorithms</td>
<td>28</td>
</tr>
<tr>
<td>The trials and the results</td>
<td>32</td>
</tr>
<tr>
<td>CHAPTER 5 CONCLUSION</td>
<td>38</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>41</td>
</tr>
<tr>
<td>VITA</td>
<td>44</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

My greatest thanks goes to Dr. Kazem Taghva for chairing my committee and advising this research. For this and for being generous with his time when I needed it I am deeply indebted to him.

I also thank my other committee members Dr. Tom Nartker, Dr. Wolfgang Bein, and Dr. Henry Selvaraj, who made substantial contributions from the perspectives of their field.

I would like to thank the faculty of the Computer Science Department and Information Science Research Institute(ISRI) of University of Nevada, Las Vegas for the formal education along with generous financial support.

Ron Young, senior software design engineer in ISRI, provided help in numerous ways in this research and he deserves special thanks. I am also quite grateful to my other colleges from ISRI, Julie Borsack, Steve Lumos and Allen Condit for the advise they gave me during the experiments that we were performing. They made their software available to me for these experiments and aided me in its use. Doina Bein helped me solve the mysteries of Latex.

I am very grateful to my mother Mence Kirovska for her inspiration and encouragement. Finally and most importantly, I thank my husband, Phillip, for his love, understanding and support.
CHAPTER 1

INTRODUCTION

The increasing ability to generate information in electronic form is probably the most remarkable manner of the late 20th century and the beginning of 21st century technology. Recent technology provides moving capabilities to store and retrieve information, to handle and distribute it effectively.

A common misinterpretation in the 1980's was that increasing use of computers would lead to decrease the use of paper. On the contrary, that led to a significant production of document generation. Each day, United States government agencies and businesses produce more than a billion pages of computer printouts, letters, photocopies and paper documents. In order to benefit from modern technology, processed information of interest must become obtainable in electronic format.

Nowadays, broad human resources are involved in the process of converting data from paper to electronic form. Nevertheless, there are many disadvantages of manual data conversion. Manual input of printed documents into a computer is very time-consuming and erroneous, and therefore cost-intensive. To diminish cost for converting data into electronic form, reading machines have been developed for many years. Text recognition systems are of immediate necessity for efficient conversion of the massive amounts of printed documents. The prime task of text recognition systems is to convert images of printed characters into identifying codes such as ASCII or EBCDIC. However, the expansion of these machines has been slower than expected and that is primarily due to deficiencies in recognition accuracy. Current text recognition systems achieve 100% accuracy only for shape-optimized fonts, but can have significantly lower accuracy for regular documents.
In this thesis we discuss the methods for finding errors in optically scanned text. Chapter 2 gives some background information on OCR, the errors pertaining to OCR and the reason for producing errors, task related to computer-based managing of textual information such as text categorization and the various methods for building the text classifier.

In chapter 3, we report on the factors that cause OCR errors and how this problems are resolved in different applications, we discuss the recent research in the field of text classification, and finally we explain the experimental environment that we are using.

In chapter 4, we describe the experiments performed, the methods used, and the decision made. We give details on lexicon-based error correction systems. Subsection 4.1 deals with creating the categories, designing the dictionary that we will use for error correction. Also it explains the similarity measures used to find a best replacement for a misrecognized word and the implementation of the error correction algorithm. Subsection 4.3 presents the experiments and the results.

Chapter 5 is the summary of the thesis.
CHAPTER 2

PRELIMINARIES

2.1 OCR

Optical character recognition (OCR) technology provides an important substitution to manual data conversion. Recognition of printed and handwritten characters has commonly been the most substantial goal of optical character recognition systems. Early systems were generally designed to recognize characters in one font and one size for example, typewritten text. Such systems depended upon invariable spacing of characters to segment individual characters [20]. OCR systems and the features used in such systems have become progressively more sophisticated as a result of extensive experimentation using several test databases that have become available in the recent years [17, 10]. Today's systems are mostly omnifont recognition systems. OCR devices are currently the most practical means of converting enormous amounts of data into electronic form. This conversion is conceptually done into the following steps [29]:

- The first step in converting a paper document into an electronic form is page scanning.

- The second step, called recognition, consists of two different technologies: zoning and text recognition.

  Scanning refers to the digitization of an input page into an electronic image. This is done through optical scanner, which senses digressions in light intensity to characterize the gray levels into either 1's for black or 0's for white. The digitized page binarized into a matrix of 1's and 0's is called bit-mapped image.

  Recognizing a page, particularly a heterogeneous one that someone could find in a magazine, is a somewhat complex process. Zoning is block segmentation function that
decomposes an image page into structural regions. These regions can either be graphic material or text. Furthermore, if the page is formatted with multiple columns (newspaper style) or any other way rather than usual manuscript format, then the reading order of the text should be maintained. Zoning can be performed either manually or automatically.

Text recognition is interpretation of text regions to their ASCII representation. Each character is isolated and converted to its ASCII equivalent. The objective in the process of segmentation is to identify each individual character within words and sentences. However, due to noise, characters may be incorrectly joined or broken. Also, because of fonts where the characters touch, there may be no easily identified location for separating characters. Recognition is then performed on each candidate character. If recognition confidence is low, in most modern OCR systems there is repeated iteration between the segmentation and recognition steps until the suspect character's recognition confidence is high. An alternate approach to character recognition is to identify whole words at once instead of individual characters. The fundamental advantage of this approach is that it prevents the need for character segmentation. Segmentation of words is much more reliable.

The following factors contribute to the difficulty of segmentation: character breakup, character touching, font style, stray marks [24].

Character Breakup Poor paper-print quality of the document, poor scanning and resembling factors can cause small breaks in the characters. Even those insignificant breaks can cause segmentation problems such as misinterpreting the single character \( m \) like \( mn \) in a small font.

Character Touching Smearing from the reproduction process can cause character to touch. The previous induce multiple characters to be recognized as single character such as \( nn \) to \( m \).

Font Style In some italic fonts, the slant of the characters is so large that successive characters may overlap each other in the vertical direction.

Stray Marks Underlining and highlighting can make segmentation very error-prone.
Most of the applications where OCR can be useful require a high degree of accuracy. In financial applications for example, misreading a digit could have serious consequences. In mail sorting, a misread address could cause a letter to be sent to the wrong location. There is also no practical method of circumventing OCR accuracy requirements in processes such as document retrieval and document classification. Error-prone documents could be easily classified incorrectly or could not be found in the process of retrieval. Currently, there are no effective techniques for eluding these accuracy requirements.

By almost any measure, the amount of data generated is increasing faster than the ability of end users to find, absorb and use this data. Due to constant increase of documents in electronic format, consumers need flexible ways of finding information of interest, and investigating it in its original or an abbreviated format. For some types of data, this can be straightforward. For example, spreadsheets, database software and statistics packages have made user access to variety of numeric data fairly comfortable. It is more difficult, however, to provide practical access to textual data, including electronic mail, newspapers, magazines, manuals, memos, technical articles, books, and many others forms of text. One of the methods for more efficient content-based text processing is text classification. Text classification deals with the assigning the contents of documents or parts of documents to one or more groups.

2.2 Text categorization

Classification is an ambiguous word in information retrieval, psychology, applied statistics, and other sciences but nearly always refers to the process of grouping items [13].

Text classification is a generalized term that includes numerous distinct information retrieval tasks, but which all deal with grouping of textual entities. In this section, we describe one such content-based management task that has received significant status in information science: text categorization. In the 90's, with the booming generation and consequential overload of electronic documents, automated text categorization has witnessed a renewed interest. Text categorization can be used in various applications, for instance automated
document indexing based on controlled vocabulary, text filtering, automatically classifying Web pages or sites, and mainly in any application that requires document organizing or selective dispatching.

Text categorization is the automated labeling of texts with thematic categories from a predefined set, rather than categorizing them in response to user request or query. An essential function of text categorization is to index documents to aid information retrieval and generate document representatives. Text categorization systems are making effort to duplicate human categorization judgments.

There are two distinct approaches to text categorization [25]. The first approach to building a text categorization system is rule-based, that is manually defining set of logical rules on how to categorize documents under given set of categories by means of knowledge engineering techniques. The other approach is machine learning approach. Within this approach, a set of documents is manually assigned to categories and then an inductive learning process is used to automatically assign categories to future documents, based on the words they contain. The advantage of the last is that can save substantial human effort. Furthermore, in terms of effectiveness achieve impressive levels of performance, making automatic categorization a qualitatively and economically feasible alternative to manual categorization.

Text categorization process consists of four phases [13]:

**Indexing** Documents are converted into text expressions occasionally called document representatives. Speed of indexing is an important matter, since large number of new documents may need to be categorized in real time.

**Categorizer Formulation** Text categorization requires a rule, hypothesis or a model that describes how document ought to be classified based on its document representative. Such internal specifications are called categorizer. Essential expert effort or complex statistical analysis might be uses in building the categorizer.

**Comparison** A binary decision is necessary, in almost all text categorization system, from each categorizer about each document. A document could, also, be compared to
number of categories at once, and convenient decision is made relying on dependencies among categories.

**Adaptation** Feedback is important role in text categorization systems. For instance, when a categorization system is built, there are usually large numbers of documents accessible that are previously manually assigned. By using feedback, the categorizer is able to incorporate the new documents without requiring manual intervention.

As specified earlier, the machine learning strategy relies upon the information from the initial set of categorized documents. That implies that the initial set comes with a correct decision matrix. A value of 1 in the decision matrix is perceived as a confirmation from the expert that a document is correctly assigned to a specific category. In addition, that is also considered as a positive example. A value of 0 in the decision matrix is interpreted as negative example and indicates the expert’s decision that the document does not fit in a particular category. During the building of the categorizer a method to test for determination of categorization effectiveness is necessary. To accomplish this testing, the initial set of documents is divided into two sets, which are not necessarily balanced: training set and test set.

*Training set* is the set of documents used to build the decision matrix to separate the documents into the different categories. By learning from the training set an “ideal document” is formed using the decision matrix for each category. These ideal documents are known as *categorizers*.

The *test set* is used for evaluating the effectiveness of the derived categorizers. The categorizer decision is compared to the ones of the expert. The similarity of the obtained values will provide the effectiveness performance of the categorizer.

A category can either be of binary or multiple type. A *binary category* refers that a document is assigned to only one category and is or is not a member of a specific category. When *multiple categories* are used, a document could have a varying degree of membership in the categories, it might be assigned to all, some, or none of the categories. Many
text categorization systems consider a text as a "bag of words", disregarding the original formatting of the text and considering the presence or absence of each word as a binary feature.

In the process of indexing, a particular text representation should uniformly be applied to the training and the test documents. Each document is usually represented by a vector of \( n \) weighted-index terms that occur in the document. One of the strategies for text representation is to identify terms with all the words occurring in the document (the "bag of words" approach). There are other more complex ways of text representation, but experiments have given discouraging results [8, 4]. Another strategy is where the weights range between 0 and 1 [6]. In rare cases, binary weights are used, where 1 denotes presence and 0 absence of the term in the document. In case of non-binary indexing, most of the time, the standard \( tf \times idf \) [8] weighting function is used, which is defined as

\[
    tf \times idf(t_k, d_j) = f(t_k, d_j) \times \log \frac{|D|}{f(t_k)}
\]

Where \( f(t_k, d_j) \) denotes the number of times term \( t_k \) occurs in document \( d_j \), and \( f(t_k) \) denotes the number of documents in which term \( t_k \) occurs at least once (document frequency of the term). This function incorporates the idea that the more often a term occurs in a document, the more representative of that document, and the more documents the term occurs in, the less disregarding it is. In this thesis term and word are used interchangeably.

Even though, \( tf \times idf \) is by far the most commonly used method, different indexing techniques have been used, especially when the training set is not available from the beginning of the process of building text categories.

2.2.1 Dimensionality reduction

High dimensionality of term space in text categorization may cause different problems. In practice, there are various methods for decreasing the dimension of the vector space. These methods, known as dimensionality reduction techniques, tend to improve the performance of the categorization system. Furthermore, dimensionality reduction is contributive, since it intends to decrease the problem of overfitting [19]. There are two distinct methods for dimensionality reduction [25]:

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Local dimensionality reduction refers to the process of choosing terms for each category. Theoretically, that would mean that for each document there is a different representation for each category.

Global dimensionality reduction refers to the process of choosing terms that will represent all categories.

2.2.2 Methods for construction of a classifier

There are different methods for inductive construction of text classifier. Classifiers are algorithms that embody a distance function that allows determination of the most likely words that will represent a class. A variety of mathematical procedures have been applied to this problem. Some of the most commonly encountered techniques are the following:

2.2.2.1 Probability text classifier

This method is based on the traditional information retrieval techniques. The training documents are used to build the categories. The system uses similarity measures between the new documents and the categories in order to classify them properly [33]. Each document, and each categorizer, can be represented as a vector of form \((\omega_1, \omega_2, ..., \omega_{|V|})\), where each component \(\omega_t\) of this vector represents the weight of the term \(v_t\) in the document and the categorizers. Each component \(\omega_t\) can be either 0 or 1. In this case, the weight represents the presence or absence of the term \(v_t\) in the document. This weight though can carry more information such as the frequency of the term in the document. The set of words in a lexicon can be represented as a vector \(V = (v_1, v_2, ..., v_{|V|})\).

Now, let \(C = (c_1, c_2, ..., c_{|V|})\) and \(D = (d_1, d_2, ..., d_{|D|})\) be sets of categories and training documents, respectively. Each category \(c_j\), is represented by a vector of the above form, where the weight, \(\omega_t\), is calculated from using term frequencies based on the training set of the documents. Using the naive Bayes assumption that the probability of each word occurring in a document is independent of the occurrence of the other words in a document, then these weights can be easily calculated.
There are two distinct versions of the naive Bayes categorizer. The first one is the Bernoulli method. In the Bernoulli method, the frequency of the words, do not play any role. Hence, each document is represented by a vector of the form $d_i = (B_{i1}, B_{i2}, ..., B_{i|V|})$, where each $B_{it}$ is either 1 or 0. In this case, the weight of each component of the categorizer $c_j$ is calculated using the following formula:

$$P(w_t | c_j) = \frac{1 + \sum_{i=1}^{|D|} B_{it} P(c_j | d_i)}{2 + \sum_{i=1}^{|D|} P(c_j | d_i)}$$

In other words, the weight of the term $w_t$ given category $c_j$ is obtained by dividing the number of documents containing the term $w_t$ and in the category $c_j$.

Now, the probability of a new document $d_i$ belonging to category $c_j$ is calculated by the following formula:

$$P(d_i | c_j) = \prod_{t=1}^{|V|} (B_{it} P(w_t | c_j) + (1 - B_{it})(1 - P(w_t | c_j)))$$

In the second method, the multinominal model, the frequency and the length of the document play a role. In this setting, a document $d_i$ is represented with a vector of the form $d_i = (N_{i1}, N_{i2}, ..., N_{i|V|})$, where $N_{it}$ is the frequency of the term $v_t$ in the document $d_i$. If we use the notation $|d_i|$ for the length of the document, then the following formulas represent the corresponding calculations for the multinominal model.

$$P(w_t | c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{it} P(c_j | d_i)}{|V| + \sum_{i=1}^{|D|} \sum_{t=1}^{|V|} N_{it} P(c_j | d_i)}$$

$$P(d_i | c_j) = P(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{P(w_t | c_j)^{N_{it}}}{N_{it}!}$$

2.2.2.2 Decision tree classifiers

A decision tree text categorizer consists of a tree in which internal nodes are labeled by terms, branches coming out from them are labeled by tests on the weight that the term has in the representation of the test document, and leave nodes are labeled by categories. Such categorizer assigns a test document by recursively testing for the weights that the terms labeling the internal nodes have in the representation of the particular document, until a leaf node is reached; the label of this leaf node is then assigned to the document. The most frequently used are binary decision trees that assume binary document representation [14].
More detailed explanation on this and the following techniques used in categorization can be found in [25].

2.2.2.3 Decision rule classifiers

*Decision rule* categorizers use an inductive rule learning method in building categorizers. Rule premises refer to the presence or absence of terms in the test document, while the rule head refers to the decision whether to assign it or not to a certain category. Rule induction methods intend to select the "best" rule from all the possible, regarding to some minimality criterion.

2.2.2.4 On-line linear classifiers

*Linear categorizers* rely on the extraction of explicit prototypical document of the category from the training set. Two different methods are used to build a categorizer: *batch induction* method and *on-line induction* method.

The first builds a categorizer by gathering information from the training set all at once. On the contrary, the other builds a categorizer after analyzing only the first training document, and incrementally refines it as it examines new documents.

2.2.2.5 Rocchio classifier

The *Rocchio categorizer* relies on an adaptation to the text categorization case of Rocchio's formula for relevance feedback in the vector space model, and it is perhaps the only text categorization method whose roots lie exclusively in the information retrieval tradition rather than in the machine learning one.

2.2.2.6 Neural network

*Neural network* categorizer is a network of units, where the input usually represent terms, the output units represent various categories of interest, and the weights on the edges that connect units represent conditional dependence relations. For categorizing a test document, its term weights are assigned to the input units. The activation of these units is propagated forward through the network, and the value that the output units take up as a
consequence determines the categorization decisions.

2.2.2.7 Example-based classifiers

*Example-based* categorizers do not build an explicit, declarative representation of the category of interest, but "parasite" on the categorization judgments that the experts have given on the training documents similar to the test documents. These methods have thus been called lazy learning systems, since they defer the decision on how to generalize beyond the training data until each new query instance is encountered.

2.2.3 Effectiveness measures

In order to say that a text representation is performing better than another, it is necessary to prove that the outcome of corresponding text categorization experiment is more effective than the other. Knowing that a set of $n$ documents has been previously categorized by binary text categorization system, and also by an expert it is easy to measure the effectiveness of the categorizer. We can specify the relationship between the system decisions and the expert judgments in so called contingency table (Figure 2.1) [3].

```

<table>
<thead>
<tr>
<th></th>
<th>expert says yes</th>
<th>expert says no</th>
</tr>
</thead>
<tbody>
<tr>
<td>system says yes</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>system says no</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>a + c = r</td>
<td>b + d = n - r</td>
<td>a + b + c + d = n</td>
</tr>
</tbody>
</table>
```

Figure 2.1: A set of $n$ classification decisions represented as a contingency table

Each entry in the table specifies the number of decisions with the specified result. For instance, $a$ is the number of times the system decides that the categorization is correct, and was also marked as correct by the expert. The effectiveness measures are defined in terms of the contingency table and there are three substantial measures of the effectiveness of the categorization system:

- **Recall** = $a / (a + c)$
- **Precision** = $a / (a + b)$
- **Fallout** = $b / (b + d)$
Recall measures the proportion of the correct documents that the system assigns to the category. Perfect recall is achieved by a system that assigns every document to a category, thereby obtaining recall of 1. On the other hand, precision is defined as a proportion of documents assigned to the category that are really members of the category. Ideal system would have precision of 1. Fallout is an alternative to precision and is defined as the proportion of the documents that are incorrectly assigned to a category by the system. Perfect precision and fallout can be achieved by a system that assigns no document to a category.
CHAPTER 3

BACKGROUND

3.1 OCR errors

Peculiarities of handwriting are so complex that even humans have difficulties in interpreting such data. Even typed materials can show a large variety of complicated factors, including [24]:

**Character Diversity** (stylized, script, bold, italicized, foreign language, etc.)

**Specialized Symbols** (technical notation, diacritical marks, etc.)

**Poor Media Quality** (opacity variations, density variations)

**Extraneous Marks** (underlining, highlighting, marginalia, stamped data).

In addition to the imperfections that may exist in the original copy of the document, processing equipment can also produce a large variety of deformities. In general, most documents combine several of these problems, from page to page and within a page. Therefore, optical character recognition (OCR) technology encounters very complicated pattern recognition problem. Fuzzy matching technics can retrieve corrupted words. The most difficult recognition problems are those in which type fonts of multiple sizes and styles occur on a single page. This is the case of most modern applications.

The main cause of recognition errors is graphical similarities. The image is not clear enough to make it possible for the OCR device to draw the right conclusion. Two things can happen: the OCR device can not classify the character at all and outputs a default character (usually ₁), or the characteris wrongly recognized, the OCR device chooses a similar shaped
character that is more highly frequent in the language. Example: argument, mountain, system, the.

Although, optical character recognition is a technology of very extensive applicability and with substantially immense economic impact only a very small part of this potential has been used. This primarily, is due to limitation on the accuracy of OCR systems. The degree of generating OCR errors also is insidious. OCR errors are just unavoidable and there is no guarantee of correct interpretation of a word even if it appears several times in a given document. With good paper originals, OCR can achieve 99% of the characters correctly recognized but there could still be 25 misspelled words per page [30]. The accuracy requirements are very demanding. OCR errors in individual words can often be corrected since they contain redundancy. Depending on the application of the optically scanned text, large post-processing efforts usually are required. This makes the need for good tools of spell checking and correction strong and urgent. The purpose of implementing different tools is to improve the correctness of the text for subsequent retrieval.

Recognition errors either can be classified by the processing steps where the error occurs or by the effect the error has on the recognized text [26]. Every spelling correction method comprises four components:

- **Spelling-error model** It describes how words can be garbled by a specific input method.

- **Dictionary** This component introduces all the correct words that are acknowledged by the spelling corrector. More accurately is for a dictionary to contain as many words as possible, so that substantially small number of words are marked as misrecognized.

- **Candidate word generation** In order to correct garbled words it is secure for the selected dictionary words to be the correct words for the candidate list.

- **Candidate word ranking** The candidate words have to be ordered so that the most likely correct candidate word is ranked highest. This requires an algorithm to compute the similarity measure between the misspelled word and each candidate word.
Most of the traditional spelling correction systems perform simple four-step procedure: *insertion, deletion, substitution, and transposition* of all the characters according to the single error model of Damerau [2]. This model is reported to cover over 80% of all mis-spellings. Accordingly, Damerau's model does not cover all recognition errors. However, there is another model proposed, that is superset of the previous, which consists of four error categories [26]:

**Case error:** a word has been correctly recognized except for the case of some of the characters. For example: *comPuEr* instead of *computer.*

**Single error:** up to three adjacent characters have been garbled at one error position in a word. For example: *iiiountain* instead of *mountain,* where *m* is recognized as *iii.*

**Multiple error:** several single errors have occurred in a word. For example: *mtonrriation* instead of *information.* Here *m* is recognized as *m,* *f* is recognized as *t,* and *m* is recognized as *iii.*

**Real word error:** a word is recognized as correct word in the language but not in the original text. For example: *he* instead of *the.*

An overview of the error groups with examples is shown in the figure below:

<table>
<thead>
<tr>
<th><em>Error group</em></th>
<th><em>error example</em></th>
<th><em>correct word</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing word</td>
<td>argue</td>
<td>argument</td>
</tr>
<tr>
<td>Truncation</td>
<td>comPuter</td>
<td>computer</td>
</tr>
<tr>
<td>Inserted Upper case letter</td>
<td>type5</td>
<td>types</td>
</tr>
<tr>
<td>Inserted digit</td>
<td>character</td>
<td>character</td>
</tr>
<tr>
<td>Inserted special character</td>
<td>mountain</td>
<td>mountain</td>
</tr>
<tr>
<td>Single error</td>
<td>inter va l</td>
<td>interval</td>
</tr>
<tr>
<td>Split Word</td>
<td>dueto</td>
<td>due to</td>
</tr>
<tr>
<td>Missing space</td>
<td>water</td>
<td>wafir</td>
</tr>
<tr>
<td>Multiple error</td>
<td>person-name</td>
<td>person_name</td>
</tr>
<tr>
<td>Special terms</td>
<td>u.jcidsq</td>
<td>words,</td>
</tr>
<tr>
<td>Impossible</td>
<td>following,</td>
<td>following:</td>
</tr>
</tbody>
</table>

Figure 3.1: Error groups with examples
3.1.1 Tools for OCR error correction

Considering the great variety of errors there are many post processing systems developed. At UNLV's Information Science Research Institute a document processing system called MANICURE (Markup ANd Image-based Correction Using Rapid Editing) was built [31]. It provides integrated facilities for creating electronic form of printed materials and has specific modules that deal with automatic detection and correction of OCR errors.

The design of MANICURE is such that it can use the characteristics of a document (i.e. word forms, geometric information about the objects in the document, and font and spacing between textual objects) to determine the logical structure of the document. Furthermore, the system can automatically detect and correct OCR spelling errors by using dictionaries, approximation matching, standard OCR errors, and frequency and distribution of words and phrases in the document. The purpose of designing MANICURE was primarily to convert printed materials from text collections into a suitable format for information retrieval applications. MANICURE consists of four modules and OCR front end that are described below:

The parser (doc_parse) is an OCR dependent module, which extracts necessary information from the output of the OCR device to build a physical representation of the document in the form of hierarchical tree. The leaves of the tree are labeled by the recognized text strings together with font information and the location of the string on the image. The nodes of the tree keep track of lines, zones and pages of the input document.

The logical document tagger (autotag) is the most practical and explicit part of the MANICURE system. Considering the physical representation produced by the parser, autotag constructs a tree representing the logical structure of the document. The leaves here are labeled by the words that form the content of the document. Characters such as end of line hyphenation are managed in this module and document structures (for instance running headers and footers) may
be removed. The nodes contain information for sentences, paragraphs, sections and section titles. In addition, autotag selects and marks structured data such as title and author.

The post processing system (ppsys) is an automatic error detection and correction program designed for OCR text. The system performs error correction through approximation matching. It uses the device’s most common confusions, and gathers structural information from the complete document. Taking into account, that the complete document has already been processed, the information of its content can be used to correct it. The post processing system builds an inverted file from the output of autotag consisting of the document’s words, their frequencies and their similarities to other words in the document. In order to mark each word as correctly recognized or misspelled, special recognizers, lexicons and dictionaries are employed. Based on that, statistical phrases are generated to correct misspellings, followed by approximation matching using word frequency information and OCR error information. Evaluation results have indicated that the correction rate of the post processing system is in the range of 15% to 50% of all misspellings in the document.

The semi-automatic user interface (rummage) is used for inspection and correction of OCR errors and markup. The part of the misrecognized words that may not been corrected by the post processing system and the other modules, can be easily corrected using rummage. Considering that the post processing system tags each misspelling, rummage can run through the document and highlight each misrecognized word. The user can then correct the misspelling by using one of the following three options: selecting the correct misspelling from the given list by a simple click of the mouse; retype the correct word; accept the word by clicking next.

Another program for OCR correction, OCRSpell, was also designed at ISRI [34]. OCR-Spell is a tool for preparing large collections of documents for input to a information re-
treival system or for presentation. The primary intention was for the system to work in conjunction with the MANICURE processing system. OCRSpell includes many OCR error-correcting techniques. It has an automatic system that uses information about typical OCR errors, dynamic knowledge about the document and uses it whenever the information is required. Any time when OCRSpell could not provide an appropriate candidate word for a misspelled word, the system "learns" from that confusion and adds the information to the device-mapping list for later use. In that manner the system will offer an adequate correction when the same type of confusion occurs.

Our research was in a similar direction: to build different kind of error correction program.

3.2 Experimental environment

The environment we have at the Information Science Research Institute (ISRI) lends itself to experimentation. ISRI has a fairly large set of test documents. These documents are part of a large textual repository that is being built by the Department of Energy (DOE) called LSS (Licensing Support System). Many of the documents have been recognized using commercial OCR devices. The LSS is a planned system that will capture and track documents pertaining to the site licensing proceedings of the Nuclear Regulatory Commission. Eventually this system will need to provide access to millions of documents.

We use only a part of this large and diverse collection of information. The collection we use is heterogeneous, not only in document quality and structure, but also in content. Different types of documents are included in the LSS, such as technical reports and analyses, quality assurance records, license applications, studies, plans, correspondence, etc. The document should be categorized into different topic groups based on their content [1]. Most of the documents are not originals; many are nth generation copies or copies from books where the edges have been skewed. The fonts are typical of what one would expect in such a diverse collection. Since the collection is scientific in nature, many documents contain formulas, graphs, photos, maps, and other graphics. The content also covers a broad range of topics. We used 138 document ranging in size from a single page to 1500
pages. For our experiment we use only few categories:

02.1 The Natural Systems of the Geologic Settings: Geologic System

02.2 The Natural Systems of the Geologic Settings: Hydrologic System

02.4 The Natural Systems of the Geologic Settings: Climatological and Meteorological Systems

04.1 Engineered Barrier Systems: Waste Package

12.1 Information for Preparation of a Geologic Repository Environmental Impact Statement: Environmental

12.2 Information for Preparation of a Geologic Repository Environmental Impact Statement: Socioeconomic

12.3 Information for Preparation of a Geologic Repository Environmental Impact Statement: Transportation

In the first run of our experiment we converted the documents to regular ASCII text and in the second run we used the same documents but ran them through MANICURE.

For the intention of categorization of the documents we used the BOW (Bag Of Words) text classifier [14]. BOW is a text classifier, which is a statistically based text categorization program. BOW is normally used in two steps:

1. BOW reads the documents and writes a model containing statistics for each document.

2. Using the model, BOW performs classification or diagnostics.

Before performing classification or diagnostics, BOW indexes the given set of documents in a way that it first reads the documents and archives the model containing their statistical information. The text index for the model must contain all the training data. The test set may also be read as a part of the model, or could be left out, to be read later on.
In most cases, the text data should be in plain text files, one file per document. No special tags are needed at the beginning or end of documents. The files should be contained in directories, such that all documents with the same class labels are contained in the same directory. BOW does not directly support classification tasks in which individual documents have multiple class labels.

When indexing a file, the file's stream of characters are converted into terms by a process called tokenization or lexing. By default, all alphabetic sequences of characters are processed in a manner that each sequence is changed to lower case and any term which is on the "stoplist" (a list of common words such as "the", "of", "is", etc.) are ignored.

Once indexing is performed a model have been archived to disc, BOW performs document classification. Statistics from a set of training documents will determine the parameters of the classifiers; classification of the set of testing documents will be the output.

3.2.1 The experiments

Each experimental run consists of four trails that pertain to the dimensionality reductions. As discussed, in the previous chapter, various techniques of dimensionality reduction can be applied to increase the efficiency of text categorization. Dimensionality reduction eliminates terms that do not provide substantial information to aid in determining the categories. Such terms are those that are misrecognized by the OCR device. Removal of non-informative words is commonly used technique in document categorization and retrieval to improve the accuracy of the results and to reduce the redundancy of the computation. The effects of several categorization methods on different document collections have been studied and the effectiveness has been evident in the experiments. Numerous reports have shown that removal of OCR errors through dimensionality reduction improves the accuracy of categorization [33]. Consequently, for our experiment we applied the four available dimensionality reduction techniques:
Default: no words are removed from the vocabulary.

Document Count removes words that occur in \( N \) or fewer documents. In our experiments \( N \) was set to be 3.

Occurrence Count: removes words that occur less than \( N \) times per document. \( N \) was set to be 10 for our experimental purposes.

Information Gain removes all but the top \( N \) ranked words by selecting words with the highest information gain. In our experiments \( N \) was equal to 10,000.

For the accuracy measure of the categorization, we compare the results from the automatic categorization performed by BOW and the manual categorization performed by experts and few geology students.

In the earlier work in the field of document categorization the following two probabilistic models for categorization were applied [15]:

- Multi-variate Bernoulli model
- Multinomial model

Both approaches make the naive Bayes assumption. Because of the independence assumption the parameters of each document can be learned separately. This greatly simplifies learning, especially when the number of parameters is large. Document categorization is just such a domain with a large number of parameters. The parameters of the documents to be categorized are words, and the number of words can be quite large. The comparison of the both "naive Bayes" generative models has given the following results:

Multivariate Bernoulli model usually performs better than the multinomial at small vocabulary sizes. On the other hand, multinomial performs well at larger vocabulary size, reducing on average 27% of the errors more than the multi-variate Bernoulli model at any vocabulary size. Furthermore, results from previous research show that the multinomial gives better results than Bernoulli on longer documents. Another point to take into consideration is that the multinomial model is more accurate classifier for collection of documents.
that greatly vary in length. This model consequently manages documents of varying length by extracting characteristics from each word that appear in the document. Evidence shows that the multi-variate Bernoulli model does not give satisfactory results in classification of documents of various lengths. Many researchers have shown interest in comparing the above models stating their explicit differences [5]. Previous formalizations of the multinomial model have omitted document length. Including document length is necessary because it specifies the number of draws from multinomial. In practice document length may be class dependent, and a more general formalization should regard this.

Previous work at ISRI has proven indeed that multinomial performs better than Bernoulli on longer documents. Taking into consideration all of these outcomes we decided to run our experiments only by using the multinomial method.

Recently at ISRI an experiment was performed using the same collection of documents that we are using in our experiment [32]. Results of that experiment have shown that two of the documents were not categorized properly. We presumed that the documents were not assigned to the correct category possibly due to errors in the text. We intended to implement some kind of a program for error correction and improve categorization. We came up with an idea to build a program that will attempt to correct the errors using the information extracted from the categories themselves.

Summarizing everything that we have previously stated, we can outline the procedure of our experiment:

1. Designing a spelling-correction program using information from the seven given categories instead of using a regular dictionary.

2. Categorize the corrected documents using the BOW text classifier.

3. Compare the results of the previous experiment and our latest research.
Many experiments have shown that errors do not affect average precision and recall, but may produce variation in ranking in information retrieval environment [28]. However, the effects of errors on automated text classification are more obvious [33, 32]. Spelling corrections of erroneous documents may improve categorization accuracy. Even though, many advanced text recognition systems have spelling correction components integrated in the devices, there is plenty of room for improvement.

Most spell checkers and OCR post processing systems are lexicon-based and use non-word errors (strings that are non-words in the language or proper names, acronyms, numbers, and technical terms) as an error. A list of acceptable candidate words is used to match against the words from the recognizes text. With this approach, every recognized word is verified using a dictionary, and if the word is not among dictionary words, an attempt can be made to correct it. In order to generate suggesting corrections, a similarity measures or probability scores are used. Any word that is not in the given list is marked as a possible error. This may lead to many false alarms, since a dictionary can not contain everything. Consequently, not all marked words are errors. Many correct words, proper nouns and acronyms that do not appear in the dictionary will be presented as errors by the system. Additionally this method may not find the real word errors (strings that are correct words in the language, but not the word found in the text). For example, if the word form has being recognized as for then the system would not mark this as a misspelling. Conventional text correction systems usually employ the lexicon-based approach.

It is important for any spelling correction system to have a suitable dictionary. If the
dictionary is too small, the candidate list of words for the misspellings will be limited. However, a lexicon that is too large occasionally could not detect misspellings due to dense word space [11]. Corrections are performed by isolating a word and checking its presence in the dictionary. If it is not in the dictionary, it can be transformed to one by using a simple four-step procedure with four simple operations: insertion, deletion, substitution and transposition of all the characters [21].

As described above, one of the most obvious methods for error correction and thereby improving the accuracy of categorization, is by the use of a dictionary. Our idea was similar. But instead of using an English dictionary or a domain specific dictionary we intended to exploit information from the categories themselves.

The ability to improve OCR accuracy using lexical context checking is highly contingent upon the subject. In the cases where standard text is used, lexical context checking is very effective. However, its effectiveness decreases substantially for documents containing proper nouns, acronyms, technical terms, formulas, numbers and etc. In other words it is not very valuable for documents that contain such nonstandard words, which is the case with our collection of documents.

4.1 Design of the dictionary and the categorizers

Our starting hypothesis was based on the knowledge that a vast dictionary will not always give the best results. Consequently, we designed our dictionary using only words from the categorizers. The concept behind our idea to extract words from the categorizers was that the words from the categorizers should be thematically closely related to the words found in future documents. That should be so, because the documents belong to the same large textual repository and all of the documents pertain to the related topics. Keeping this in mind, we can say that our dictionary is a limited kind of a domain-specific dictionary.

We used 138 documents of various types such as studies, reports, analyses, plans, etc. From these 138 documents, 104 were selected as a training set to build the categorizers for each category. The categorizers gathered statistical information from the terms in the documents. These terms represent a perfect document that fits in the corresponding
category. The categories are represented with the same list of words. The words in the categorizers are ranked in descending order. The ranking depends upon the relevance of the words to the particular class. In that manner, even though the categorizers contain same list of word the ordering of the words in the categorizers is different.

In our experiment, each categorizer contained 34,248 words. As we previously mention, we used BOW text classifier for document classification. Detailed explanation for BOW is given in chapter 2. BOW creates the model for each categorizer as a two column rank-ordered list, where the first column contains the words and the second the corresponding word weights. An example of the categorizers is shown in Figure 4.1.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{02.1} & \textbf{02.2} & \textbf{02.4} \\
\hline
\text{cross} & 0.006649 & \text{water} & 0.018576 & \text{site} & 0.006073 \\
\text{drift} & 0.006393 & \text{flow} & 0.011375 & \text{data} & 0.004995 \\
\text{fault} & 0.005778 & \text{level} & 0.010518 & \text{guidance} & 0.004076 \\
\text{sta} & 0.005661 & \text{data} & 0.006767 & \text{maximum} & 0.003836 \\
\text{rock} & 0.005464 & \text{mountain} & 0.005108 & \text{sites} & 0.003516 \\
\text{data} & 0.005317 & \text{quot} & 0.005052 & \text{wind} & 0.003356 \\
\text{zone} & 0.004681 & \text{yucca} & 0.004486 & \text{meteorological} & 0.003177 \\
\text{mountain} & 0.004191 & \text{altitudes} & 0.004288 & \text{document} & 0.003077 \\
\text{doe} & 0.004016 & \text{unsaturated} & 0.004194 & \text{yucca} & 0.002777 \\
\text{erosion} & 0.003606 & \text{table} & 0.003902 & \text{precipitation} & 0.002617 \\
\text{percent} & 0.003562 & \text{usw} & 0.003883 & \text{projections} & 0.002497 \\
\text{yucca} & 0.003474 & \text{unit} & 0.003798 & \text{reference} & 0.002397 \\
\text{quot} & 0.003394 & \text{repository} & 0.003713 & \text{area} & 0.002178 \\
\text{cm} & 0.003394 & \text{depth} & 0.003430 & \text{conditions} & 0.002078 \\
\hline
\end{tabular}
\caption{Examples of the categorizers}
\end{table}

Based on this categorizers, if a document contained the single word \textit{waste} it is more likely to be categorized in the category 12.1 since it has the highest weight for this word. Such examples are given in the following table:

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
\textbf{term} & \textbf{02.1} & \textbf{02.2} & \textbf{02.4} & \textbf{04.1} & \textbf{12.1} & \textbf{12.2} & \textbf{12.3} \\
\hline
\text{waste} & 0.000512 & 0.002375 & 0.000739 & 0.005936 & 0.013091 & 0.002311 & 0.005090 \\
\text{geologic} & 0.002816 & 0.000556 & 0.000479 & 0.000386 & 0.000358 & 0.000175 & 0.000309 \\
\text{tunnel} & 0.003211 & 0.000009 & 0.000020 & 0.000077 & 0.000314 & 0.000062 & 0.000063 \\
\hline
\end{tabular}
\caption{Words and their corresponding weights for each category}
\end{table}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Our dictionary contains all of the 34,248 words. We used seven categories for our experiment. The documents were represented as files and the categories as directories. The categories were assigned based on ground truth (manual) assignment. Knowing the correct categorization for each document we arranged the documents in the corresponding directories (Figure 4.2).

The most recent experiments at ISRI were related to the effects that errors might have on training the classifier to build the categorizers [32]. The experiments were set that would give the most perceptiveness of those effects. There are two distinct methods in which errors can influence categorization. First, categorization can be affected by the presence of errors in the training set and second, by reducing the ability of new documents to be categorized correctly. Accordingly, four experiments were conducted.

Good Training/Bad Test Set
Mixed Training/Mixed Test Set
Good Training/Auto-Corrected Set
Good Training/Manually-Corrected Set

It was discovered that errors could cause improper categorization for the new documents. As a matter of fact, two documents were not classified properly in the first, second and fourth experimental run. The first experiment was set such that the training set was selected from the set of good quality documents. The test set was selected from the set of poor OCR quality documents. In the second trial the training and the testing set consist from both good and bad documents. In the third run, one of the documents that previously did not categorize properly, was correctly categorized. This experimental run was exactly the same as the first trial, but the two poorly recognized documents were first run through MANUCURE. The fourth run was same as the third except the two documents were manually corrected.
4.2 The algorithms

Our intention was to make an effort to correct the two documents using the information from the categorizers. In that direction, we designed a program for error correction using the introductory premises. Description of our program follows.

We were mostly concentrated on the error finding process. First of all we needed to decide what are the errors in the document. The errors can be determined by spell checking of isolated words. Accordingly, we implemented a program for spell checking and we were finding all so-called misspellings in the document using a broad dictionary. An individual word is presumed to be accurately recognized if it is in the dictionary. Otherwise, it is practically misrecognized. Those words that are found in the dictionary are discarded and only misspellings remain. Knowing which words are potential misspellings we can then use our dictionary in order to correct the doubtful words.

Our spelling correction program was designed to handle case errors, single and multiple errors according to the model proposed by Takahashi [9]. The words with the greatest possible similarity according to some measure can be picked from the dictionary as candidate words. If there is only a single word with the greatest possible resemblance, the garbled word is corrected automatically. To generate candidate words efficiently, the spelling correction program compares the resemblance between each dictionary word and each misspelled word. The candidate word list comprises all the dictionary words that have some characters in common with the garbled word.

Different methods for computing similarity measure are used to generate the correction list. For example, noisy channel models [12], edit distance combined with word frequency [16], and confusion sets [7] are the most commonly used. In our program we integrated the edit distance function as a similarity measure. [22]

Following Wagner and Fisher [22], let $\Sigma$ be a finite alphabet. An edit distance operation is a pair of strings $(a, b) \neq (\epsilon, \epsilon)$ over $\Sigma$ of length less than or equal to 1. String $x$ results from $w$, represented with $w \rightarrow x$, if there are strings $\alpha$ and $\beta$ such that $x = \alpha b \beta$ and $w = \alpha a \beta$. $(a, b)$ is a substitution operation if $a \neq \epsilon$ and $b \neq \epsilon$, a delete operation if $b = \epsilon$ and $a \neq \epsilon$, and an insertion operation if $a = \epsilon$ and $b \neq \epsilon$. We use the term edit distance to
refer to some combinations of substitution, deletion, or insertion [27].

The function `word_check` calculates the edit distance between a misspelled word and each dictionary word, and returns the one with the minimum edit distance over all. The edit distance function defines the similarity between two words as the minimum cost if the two words are compared character by character. The more similar the words are, the lower their distance score is. The score is proportional to the number of substitution, deletion or insertion operations performed on the misspelled word. Significant constraint that we have taken care of in the `word_check` function is that for various length words different edit distances were calculated. This criterion ensures that the generated candidate word list is small so that the words are the most affiliated to the misspelling. The large candidate word set mean that for a garbled word too many and too vaguely spelling suggestions are offered if no additional restrictions are used. If we consider for instance an edit distance with value of 3 for the word of length 3 then that is absurd. The justification is that the replacement could possibly be totally irrelevant to the initial misspelled word. The candidate word would then be completely unrelated to the misspelling. To determine appropriate edit distance values for each word taking in to consideration the length, we applied the following function:

\[ n = \text{ceil}(\text{word.length} / 3) \]

Often could happen that two candidate words competing for replacement of a garbled word, to have equal edit distances. In that case, another measure needs to be considered. We consider the weights of the words of the category. From the ground truth documents we know the correct category for each of the documents. Accordingly, we have created a file `correct_cat` which has information on the correct categorization of the 34 documents from the test set. Word weight information for each category is held in the `weight` directory, that we have also created for the experimental purposes. At this point we look up where the current document belongs in the `correct_cat` file. For possible replacement words with equal edit distance, the word weights determine the new candidate word. We compare the weights for the two equal edit distance words in the corresponding category. The winning word that becomes the possible replacement is the one with a higher weight. Then the
procedure continues in to finding a better replacement. The objective is to discover from
the dictionary a candidate word that has lower edit distance than the current. If none is
found, then the last candidate replaces the misspelling. The process proceeds until all the
misspellings from the current document are checked against the dictionary and potential
replacement is retrieved. Example of this procedure is follows:

test_file: lend watar land watar water mountain, functon, meefing, meeing meeting.

misrecognized words: watar watar functon meefing meeing

The allowed maximum edit distance for the word watar is $d = 2$

The competing words are:

- watar
  - weight1 = 0.000000
  - weight2 = 0.006767

data edit_dist = 2

water edit_dist = 1

The competing words are:

- water
  - weight1 = 0.018576
  - weight2 = 0.000019

watar edit_dist = 1

watar water min_edit_dist = 1

The allowed maximum edit distance for the word functon is $d = 3$

function edit_dist = 2
The competing words are:

\[
\begin{align*}
\text{function} & \quad \text{functions} \\
\text{weight}1 &= 0.000443 & \text{weight}2 &= 0.000226 \\
\text{function} & \quad \text{edit\_dist} = 2 \\
\text{function} & \quad \text{min\_edit\_dist} = 2
\end{align*}
\]

The allowed maximum edit distance for the word meefing is \(d = 3\)

The competing words are:

\[
\begin{align*}
\text{meefing} & \quad \text{meaning} \\
\text{weight}1 &= 0.000000 & \text{weight}2 &= 0.000009 \\
\text{meaning} & \quad \text{edit\_dist} = 3 \\
\text{meeting} & \quad \text{edit\_dist} = 2
\end{align*}
\]

The allowed maximum edit distance for the word meeining is \(d = 2\)

The competing words are:

\[
\begin{align*}
\text{meeting} & \quad \text{meetings} \\
\text{weight}1 &= 0.000019 & \text{weight}2 &= 0.000009 \\
\text{meeting} & \quad \text{edit\_dist} = 2 \\
\text{meefing} & \quad \text{edit\_dist} = 1 \\
\text{meefing} & \quad \text{meefing} & \quad \text{min\_edit\_dist} = 1
\end{align*}
\]

The allowed maximum edit distance for the word meefing is \(d = 2\)

The competing words are:

\[
\begin{align*}
\text{meefing} & \quad \text{meaning} \\
\text{weight}1 &= 0.000000 & \text{weight}2 &= 0.000009 \\
\text{meaning} & \quad \text{edit\_dist} = 2
\end{align*}
\]
The competing words are:

meaning       mining
weight1=0.000009 weight2=0.000038

mining edit_dist=2
meeting edit_dist=1

The competing words are:

meeting       meetng
weight1=0.000019 weight2=0.000009

meeting edit_dist=1
meetng        meeting min_edit_dist=1

Currently, the program cannot deal with split errors, since it is using the space as a delimiter. Usually a split error results in two or more invalid strings, consequently the program will mark the strings as potential errors. However, it is not possible to correct the split error simply by eliminating the space.

4.3 The trials and the results

Summarizing all that we have explained, the experiments were made to test the effect of the spelling correction to the documents. For that intention we took into consideration the two documents that were not categorized properly in the previous experiment. From the prior experience, we determined our trials in the following manner:

Good Training/Bad Test Set

Good Training/Auto-Corrected Set

In each experimental trial, we perform several distinct runs based on the multinomial probability model. In addition, each trial includes a limited vocabulary run that employs the multinomial probability techniques. The limited vocabulary merges several dictionaries. It includes also domain specific terms that an elementary dictionary may have neglected in the
process of indexing. The vocabulary consists of 413,216 words derived from several general
dictionaries, geologic and radiologic specific dictionary, and specific thesauri from LSN. In
the process of indexing, in both information retrieval and categorization, it is beneficial
to restrict the vocabulary to pre-defined control terms. This principle is commonly used.
We applied, also, all the dimensionality reduction techniques described in chapter 3. The
accuracy rates for each dimensionality reduction is shown in Table 4.2.

<table>
<thead>
<tr>
<th>Multinomial</th>
<th>Good Training/</th>
<th>Good Training/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bad Test Set</td>
<td>Auto-Corrected Set</td>
</tr>
<tr>
<td>Default</td>
<td>94.12</td>
<td>94.12</td>
</tr>
<tr>
<td>Document Count</td>
<td>97.06</td>
<td>97.06</td>
</tr>
<tr>
<td>Information Gain</td>
<td>94.12</td>
<td>97.06</td>
</tr>
<tr>
<td>Occurrence Count</td>
<td>97.06</td>
<td>97.06</td>
</tr>
</tbody>
</table>

Table 4.2: Average accuracy rates for each dimensionality reduction

The categorization experiment compares the decisions made by the system and some
standard of correctness, usually human category assignment. In our case we compare the
results between BOW text classifier and the judgment from the experts. The outcome of
the categorization is shown in Table 4.3.

The results indicate that the two documents were not categorized properly even after
the spelling correction was done. In the first run both documents were not assigned to the
correct category, as it was the case in the earlier experiment. In the second run we run
the documents through MANICURE, and then we corrected the rest of the misspellings
by our spelling correction program. At this point the document MOL.19990118.0060, was
correctly recognized, however, the other document was not.

Taking into account the results, we can say that the corrections did not make any
difference in categorizing the documents. Even though the spelling correction program
corrected on average 15% of the misspellings, the outcome was exactly identical to the prior
experiment.
Algorithm 4.3.1 Word_Check

/* We compare a misspelled word with the dictionary words and we choose
the one with the smallest edit distance:*/
{
for each misspelled word we calculate the allowed edit distance
d = ceil($strlen(misspelled_word)$);
search the dictionary for the best fit:

    take each word from the dictionary;
    {
        calculate the edit distance between the
        misspelled word and the dictionary word
        and store in the variable $n$;

        if $n$ greater than the pre-calculated allowed edit distance $d$
            do not consider those dictionary words;
        else
            {
                /* consider only that candidate word that have edit
distance less or equal to the pre-calculated allowed
edit distance */

                if $n$ less than the pre-calculated allowed edit distance $d$
                    {
                        if the calculated value for $n$ is the best
                            up to this point, store the candidate dictionary word
                            in the variable $rword$;

                        store $n$ in the variable $d$.
                        $d$ holds the minimum edit distance value;
                    }
                else
                    {
                        if there are two candidate words with the same edit distance
                            competing to replace the misspelled word,
                            we call the weight_check function;
                    }
            }
    }
}
Algorithm 4.3.2 Weight\_Check

/* Search for weights of two words and returns the word with the higher weight among them */

{

    find the correct category for the current file;

    search for the weights of the two candidate words in the corresponding category;

    compare the weights and return the word with higher weight;

}

Algorithm 4.3.3 Error\_Correction

main\_program:

{

    call the executable with a name of a file which contains the misspelled words and the dictionary file;

    for each misspelled word, the algorithm word\_check is called;

    the resulting best fit is stored in the in .tbl file that keeps track of the misspelled words as well as the result words;

}
<table>
<thead>
<tr>
<th>document_id</th>
<th>correct_category</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOL.19980729.0051</td>
<td>02.2</td>
</tr>
<tr>
<td>MOL.19981103.0084</td>
<td>02.2</td>
</tr>
<tr>
<td>MOL.19990118.0060</td>
<td>02.2</td>
</tr>
<tr>
<td>MOL.19990312.0209</td>
<td>02.2</td>
</tr>
<tr>
<td>MOL.19990604.0111</td>
<td>02.4</td>
</tr>
<tr>
<td>MOL.19980724.0394</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19980916.0395</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19980921.0029</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19981110.0430</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19981215.0031</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19981222.0137</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990511.0239</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990518.0261</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990524.0143</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990601.0140</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990615.0099</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990617.0237</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990702.0030</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990712.0195</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990713.0238</td>
<td>04.1</td>
</tr>
<tr>
<td>MOL.19990610.0347</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19971110.0454</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19981210.0240</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990127.0270</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990316.0055</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990520.0188</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990528.0311</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990604.0190</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990702.0236</td>
<td>12.1</td>
</tr>
<tr>
<td>MOL.19990517.0135</td>
<td>12.2</td>
</tr>
<tr>
<td>MOL.19990622.0159</td>
<td>12.2</td>
</tr>
<tr>
<td>MOL.19990629.0085</td>
<td>12.2</td>
</tr>
<tr>
<td>MOL.19980330.0450</td>
<td>12.3</td>
</tr>
<tr>
<td>MOV.19960903.0175</td>
<td>12.3</td>
</tr>
</tbody>
</table>

Figure 4.2: The test set of documents and their corresponding correct category
<table>
<thead>
<tr>
<th>docid</th>
<th>correct_category</th>
<th>chosen_category</th>
<th>weight</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOL.19980729.0051</td>
<td>02.2</td>
<td>02.2</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19981103.0084</td>
<td>02.2</td>
<td>04.1</td>
<td>1.000</td>
<td>incorrect</td>
</tr>
<tr>
<td>MOL.19990118.0060</td>
<td>02.2</td>
<td>04.1</td>
<td>1.000</td>
<td>incorrect</td>
</tr>
<tr>
<td>MOL.19990312.0209</td>
<td>02.2</td>
<td>02.2</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990604.0111</td>
<td>02.4</td>
<td>02.4</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19980724.0394</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19980916.0395</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19980921.0029</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19981110.0430</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19981215.0031</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19981222.0137</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990511.0239</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990518.0261</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990524.0143</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990601.0140</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990615.0099</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990617.0237</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990702.0030</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990712.0195</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990713.0238</td>
<td>04.1</td>
<td>04.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19960610.0347</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19971110.0454</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19981210.0240</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990127.0270</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990316.0055</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990520.0188</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990528.0311</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990604.0190</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990702.0236</td>
<td>12.1</td>
<td>12.1</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990517.0135</td>
<td>12.2</td>
<td>12.2</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990622.0159</td>
<td>12.2</td>
<td>12.2</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19990629.0085</td>
<td>12.2</td>
<td>12.2</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOL.19980330.0450</td>
<td>12.3</td>
<td>12.3</td>
<td>1.000</td>
<td>correct</td>
</tr>
<tr>
<td>MOV.19960903.0175</td>
<td>12.3</td>
<td>12.3</td>
<td>1.000</td>
<td>correct</td>
</tr>
</tbody>
</table>

Table 4.3: Categorization results from the first run
CHAPTER 5

CONCLUSION

Providing appropriate information in the least amount of time is a very important aspect of information retrieval science. However, it is difficult to obtain printed information in a format directly usable by an information retrieval system. Correctly generating documents in ASCII format from printed documents is a very difficult process. Optical character recognition (OCR) is an essential part of this process. The essence of the transition of printed documents into an electronic format is due to the increasing needs of users to access them in flexible ways. Consequently, different methods for data managing were developed such as information retrieval, text classification, etc. In general, all issues concerning document organization and filing, are handled by the automatic categorization technique.

Categorization of textual data into content-oriented predefined classes has gained more significant prominence in information science in the last ten years. It is a very useful technique in a variety of documents handling systems. Many methods can be used to categorize documents as soon as the words are known. Optical character recognition technology plays an important role in the process of data conversion. It is widely used as a method to move text resources from paper medium to electronic format. However, OCR can garble a large proportion of words, particularly when low quality documents are used. Unfortunately, not all documents are clean. OCR systems are regularly encountered with poor quality documents due to light originals, copies, or poor contrast, even if high-resolution scanners are used. State of the art OCR devices do not provide accurate conversion for documents of such quality [18, 23].

The problems that arise from the presence of errors in OCR produced documents sub-

38

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
stantially influence text categorization. The errors may garble crucial words and undermine weighting formulas. In order to increase categorization effectiveness post-processing of erroneous documents is often necessary.

Experiments were conducted at ISRI testing the effects of document categorization in the presence of OCR errors. We designed an error-correcting program with the intention of improving the process of categorization.

Lexical information has long been used as an aid to error-correction systems. Correcting documents using lexical checking is contingent upon the context topics. Such knowledge can be used to great advantage in resolving recognition ambiguities. Our motivation was provoked by the successful rates that lexical context checking have given in the process of error correction. Its use decreases substantially for documents that contain non-standard words, such as technical terms, proper names, numbers and acronyms. Because our document collection involves a high concentration of such terms, we created a quasi-lexicon.

Our dictionary contained words extracted directly from the training documents. It consists of 34,248 words. We used for the training set good-quality documents, but there was still a small amount of errors present. Considering this, our dictionary contained some garbled words. Some of them were also words that were not properly recognized in the process of conversion. We kept in mind these facts, but hoped that it will not greatly influence the results. A very small percent of the words that we attempted to correct were replaced with garbled words. However, most of the words were corrected. We applied our error-correcting program to all training and test documents. After that we applied automated categorization to the corrected documents. We obtained comparable results with the previous experiment. This indicates that the correction program using document information did not improve the process of categorization, but nonetheless did not worsen it. Justification for the conclusion lays in the fact that except for the two documents that were not categorized properly in the prior experiment, the rest were correctly categorized.

Summarizing all of the above, we can certainly conclude that OCR errors might not be the only reason for not categorizing the two documents properly. An important factor to be mentioned is that text categorization does not achieve 100% accuracy even with
good OCR documents. The categorizer may not be able to place a document in a correct category, no matter how clean is the document. We were persuaded in this, because some documents that contain more errors than the one that was not categorized correctly were actually assigned to a correct category.

We should point out that studies have shown even expert bibliographic indexers to disagree on essential categorization decisions [13].

There is no ideal feature that will classify all documents with 100% accuracy.
BIBLIOGRAPHY


VITA

Graduate College
University of Nevada, Las Vegas

Lidija K. Mackovski

Home Address:
4606 Wild Oak Way
N.Las Vegas, NV 89031

Degrees:
Bachelor of Computer Science, 1996
Department of Electrical and Computer Engineering
University of St.Cyril and Methodius, Skopje, Macedonia

Thesis Title: Effects of OCR errors on text categorization

Thesis Examination Committee:
Chairperson, Dr. Kazem Taghva, Ph.D.
Committee Member, Dr. Thomas Natker, Ph.D.
Committee Member, Dr. Wolfgang Bein, Ph.D.
Graduate Faculty Representative, Dr. Henry Selvaraj, Ph.D.