Robust Retrieval of Noisy Text*

Daniel P. Lopresti
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, NJ 08540
dpl@mitl.research.panasonic.com

Abstract

In this paper, we examine the effects of simulated OCR errors on Boolean query models for information retrieval. We show that even relatively small amounts of such noise can have a significant impact. To address this issue, we formulate new variants of the traditional models by combining two classic paradigms for dealing with imprecise data: approximate string matching and fuzzy logic. Using a recall/precision analysis of an experiment involving nearly 60 million query evaluations, we demonstrate that the new fuzzy retrieval methods are generally more robust than their "sharp" counterparts.

1. Introduction

When developing models and algorithms for information retrieval, it is often convenient to assume that the contents of the database are, in a sense, perfect: the text has been carefully edited and proof-read, all of it belongs, nothing is missing, and there are few if any errors. It is becoming increasingly evident, however, that the explosive growth of on-line services coupled with the informal nature of electronic communications will result in a broader range of document qualities than encountered in the past. Text taken directly from such sources (*e.g.*, Usenet newsgroup postings, WWW pages) frequently contains typographic errors, alternate spellings, non-traditional punctuation, and various sorts of spurious information.

Moreover, optical character recognition (OCR) and document image analysis (DIA) technologies have improved to the point that fully autonomous electronic filing systems are now becoming feasible. A wide variety of data can be extracted from a scanned page. While many of these attributes may prove useful for later recall, the results from OCR'ing the text are particularly important from the standpoint of retrieval.

Unfortunately, unattended OCR still has the potential to introduce significant levels of "noise" in a database. Even state-of-the-art technology produces garbled output when confronted by unanticipated text formatting (*e.g.*, shaded backgrounds, unusual fonts, white-on-black characters). Accuracy rates can drop to 80% or lower for document images that humans have no trouble reading [18].

Rather than assume such data will be excluded from digital libraries, it seems appropriate to study its impact and identify ways of mitigating the damage. More specifically, in this paper we examine the effects of simulated OCR errors on Boolean query models for information retrieval. As we shall show, even relatively modest amounts of noise can be deleterious.

We also formulate new variants of the traditional models by combining two classic paradigms for dealing with imprecise data: approximate string matching and fuzzy logic. For the most part, the fuzzy approaches are more robust than their "sharp" counterparts. All of the retrieval models are presented using the same standard notation, making it easier to compare them and judge their similarities and differences.

The remainder of the paper is organized as follows: in the next section we briefly survey previous work in the area. Section 3 describes the various retrieval models and algorithms. We discuss evaluation criteria in Section 4. In Section 5 we present experimental results involving 59.6 million queries against artificially degraded news articles taken from the Internet. Finally, we offer our conclusions in Section 6.

*Presented at the *Forum on Research and Technology Advances in Digital Libraries*, Washington, DC, May 1996.

2. Related research

There have been a number of previous studies on OCR errors and their impact on information retrieval. For example, Taghva *et al* used output from several commercial OCR packages to evaluate the accuracy of the *SMART* system, which employs the vector space model [20, 23]. They concluded that when recognition rates are high, the effects for long documents are minimal, but that for short documents under certain term-weighting schemes, retrieval accuracy decreases rapidly as the number of OCR errors increases. In other papers, they examined the *BASISplus* system [22], which is based on a Boolean retrieval model, and the *INQUERY* system [21], which uses probabilistic IR. Their conclusions in these cases were similar.

Croft *et al* used a synthetic OCR error generator to study the effects of noise on a probabilistic retrieval model [2]. These results also show that such damage can have an impact on short documents. Recently, Tsuda *et al* studied how simulated single-character substitutions can affect a vector space clustering model [24]. They concluded that the system in question may be able to tolerate relatively large numbers of errors. A similar conclusion was reached by Ittner *et al* in a study involving faxed documents [8].

Several papers have also suggested ways of coping with OCR noise in the design of retrieval systems. For example, Pearce has proposed using n-grams when building hypertext links from the output from an OCR package [17]. In a paper by Myka and Güntzer, pattern matching techniques for attempting to reduce the impact of noise are discussed [14]. Wiedenhofer presents an indexing procedure that employs character hypotheses lattices which are post-processed in various ways to eliminate suspected errors [27].

The research we report here can be distinguished from each of these earlier works in one or more ways. Unlike studies involving commercial systems that treat the IR algorithm and/or the OCR error source as a "black box," we specify both of these explicitly, making it possible to perform a more detailed comparison of the relationships between the two. The new fuzzy retrieval models we present also are quite different from those examined previously. Finally, our use of a Boolean query language allows for a richer range of inputs than papers where the search is based on a single term.

3. Retrieval models and algorithms

For the purposes of relating the traditional retrieval models with several new ones to be described later, we find it helpful to adopt a consistent, precise notation. While this approach may seem overly formal at times, it makes it possible for us to compare directly the various model components.

We begin with some definitions. A *string*, $S = s_1 s_2 \ldots s_n$, is a finite sequence of characters chosen from a finite alphabet, $s_i \in \Sigma$. String $A = a_1 a_2 \ldots a_m$ is a *substring* of string $B = b_1 b_2 \ldots b_n$ if $m \leq n$ and there exists an integer k in the range $[0, m-n]$ such that $a_i = b_{i+k}$ for $i = 1, 2, \ldots, m$.

A *query term*, T, and a *document*, D, are both strings. A *database* Δ is a finite set of documents, $\Delta = \{D_1, D_2, \ldots, D_n\}$. A *membership function* $\mathcal{M}$ is a mapping between term-document pairs and a specific set of values in the interval [0, 1]. Although the precise range of $\mathcal{M}$ depends on the retrieval model, generally it can be interpreted as the degree to which a particular document is a member of the set of all documents containing the given term. In the extreme, $\mathcal{M}(T, D) = 1$ when the term's presence in the document is strong, and $\mathcal{M}(T, D) = 0$ when the term is completely absent.

In the case of the Boolean operators AND, OR, and NOT, we distinguish between the forms that are written as part of the query (*e.g.*, (cats AND dogs)) and the functions themselves. In the case of AND and OR, the functions are $\mathcal{F}_{AND}$ and $\mathcal{F}_{OR}$ and they map tuples from the range of $\mathcal{M}$ back into the range of $\mathcal{M}$. $\mathcal{F}_{NOT}$ is defined similarly, but operates on a single value instead of a tuple.

A simple query $\mathcal{Q}$ written as (T_1 OP T_2) is "run" on a particular document D by evaluating:

$$\mathcal{Q}(D) = \mathcal{F}_{OP}(\mathcal{M}(T_1, D), \mathcal{M}(T_2, D)) \qquad (1)$$

The extension to more complex queries containing multiple operators and increasing numbers of terms is straightforward.

3.1. Boolean retrieval

Perhaps the simplest, most familiar, text retrieval model is the Boolean model. In this case, the range for the membership function is $\{0, 1\}$, and the function itself is:

$$\mathcal{M}^B(T, D) = \begin{cases} 1 & \text{if } T \text{ is a substring of } D \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

The operators correspond to the standard Boolean ones, as indicated in Table 1.

A typical Boolean query might be ((Clinton AND Gore) OR U.S. leaders), which is interpreted to mean all documents containing the substrings "Clinton" and "Gore" or the substring "U.S. leaders".

x	y	$\mathcal{F}^B_{AND}(x,y)$	$\mathcal{F}^B_{OR}(x,y)$	$\mathcal{F}^B_{NOT}(x)$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Table 1. Standard Boolean operations.

3.2. Fuzzy Boolean retrieval

A shortcoming of traditional Boolean retrieval is that it requires the terms to appear exactly in the document as substrings. If, for example, the OCR process used to generate a particular database has made the common mistake of recognizing an 'l' (lower-case 'el') as an 'I' (upper-case 'eye'), then the previous query may fail for some documents because the specific sequence of characters "Clinton" is no longer present.

Rather than insist that a query term appear exactly in the document, we might instead ask the question, "Is there anything *similar to* the term in the document?" In other words, the membership function has a continuous range:

$$\mathcal{M}(T,D) = \uparrow \begin{array}{ll} 1 & \text{if } T \text{ appears exactly in } D \\ 1-\epsilon & \text{if something much like } T \text{ appears in } D \\ \cdots & \\ \epsilon & \text{if something a bit like } T \text{ appears in } D \\ 0 & \text{if nothing resembling } T \text{ appears in } D \end{array} \quad (3)$$

Such a model can be realized by combining two classic paradigms for dealing with imprecise data: approximate string matching and fuzzy logic.

A standard measure for approximate string matching is provided by *edit distance* [9], also known as the "k-differences problem" in the literature. In general, the following three operations are permitted: (1) delete a character, (2) insert a character, (3) substitute one character for another. Each of these is assigned a cost, c_{del}, c_{ins}, and c_{sub}, and the edit distance is defined as the minimum cost of any sequence of basic operations that transforms one string into the other.

This optimization problem can be solved using a well-known dynamic programming algorithm [15, 25]. Let $T = t_1 t_2 \ldots t_m$ be the term, $D = d_1 d_2 \ldots d_n$ be the document, and define $dist_{i,j}$ to be the distance between the first i characters of T and the first j characters of D. The initial conditions are:

$$\begin{array}{lcll} dist_{0,0} & = & 0 & \\ dist_{i,0} & = & dist_{i-1,0} + c_{del}(t_i) & 1 \le i \le m \\ dist_{0,j} & = & dist_{0,j-1} + c_{ins}(d_j) & 1 \le j \le n \end{array} \quad (4)$$

and the main dynamic programming recurrence is:

$$dist_{i,j} = \min \left\{ \begin{array}{lcl} dist_{i-1,j} & + & c_{del}(t_i) \\ dist_{i,j-1} & + & c_{ins}(d_j) \\ dist_{i-1,j-1} & + & c_{sub}(t_i, d_j) \end{array} \right. \quad (5)$$

for $1 \le i \le m$, $1 \le j \le n$. When Equation 5 is used as the inner-loop step in an implementation, the time required is $O(mn)$, where m and n are the lengths of the two strings.

This common formulation requires the two strings to be aligned in their entirety. The variation we use is modified so that a term, which is relatively short, can be matched against a much longer document to locate regions of high similarity (this application is sometimes called "word spotting"). The initial edit distance is made 0 along the entire length of the document (allowing a match to start anywhere), and the final row of the edit distance table is searched for its smallest value (allowing a match to end anywhere). The initial conditions become:

$$\begin{array}{lcll} dist_{0,0} & = & 0 & \\ dist_{i,0} & = & dist_{i-1,0} + c_{del}(t_i) & 1 \le i \le m \\ dist_{0,j} & = & 0 & 1 \le j \le n \end{array} \quad (6)$$

The inner-loop recurrence (*i.e.*, Equation 5) remains the same.

Figure 1 shows the results of this computation for a simple example where we have assumed that the costs are constant, $c_{del} = c_{ins} = c_{sub} = 1$. An exact match for the term "shell" appears in the document, as indicated. This corresponds to the smallest value in the final row, 0. Also note that another, very similar substring is found at the same time ("sell").

While it is fairly common for implementations of Equation 5 to employ constant editing costs, the general way in which the algorithm is formulated is much more powerful than this. In the event that we know the distribution of the OCR errors *a priori* (*e.g.*, via a confusion matrix), we can take advantage of this fact by setting the editing costs to be inversely proportional to the frequencies of the error patterns in question. So, for example, if the substitution $e \rightarrow c$ is ten times as likely to occur as the substitution $M \rightarrow W$, its cost is made one tenth as much.

Whatever the policy regarding costs, we define $\mathcal{E}(T, D, i)$ to be the edit distance at index i in the final row of a table built in this fashion. Then $\mathcal{E}(T, D) = \min(\mathcal{E}(T, D, i) \mid 0 \le i \le n)$ corresponds to the best match(es). Under the constant cost assignment given previously, the maximum possible value in this row is m, which represents deleting all of the characters in the query term. Hence, $\mathcal{E}(T, D)$ can fall anywhere in the range $[0, m]$. We obtain a membership function

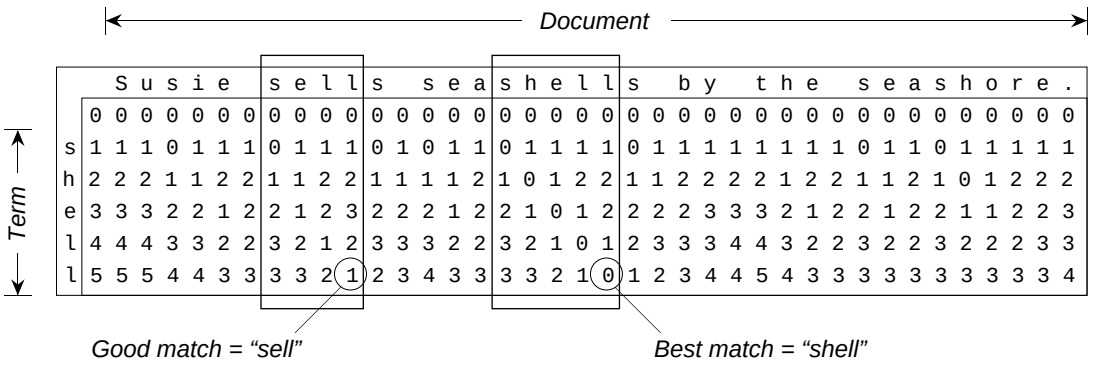

Figure 1. An example of edit distance "word spotting."

with range $[0, 1]$ for this model through the use of an exponential decay:

$$\mathcal{M}^{FB}(T, D) = \frac{1}{e^{\alpha \mathcal{E}(T,D)/(m - \mathcal{E}(T,D))}} \qquad (7)$$

where α is a constant to be determined.

Table 2 shows membership values for terms of various lengths and edit distances, assuming $\alpha = 1$. If a term appears exactly in the document, the value returned is 1.0. If none of the characters in the term appear anywhere in the document, the value returned is 0.0. These boundary conditions correspond nicely with the traditional model presented earlier.

	Query Term Length (m)					
$\mathcal{E}(T, D)$	*2*	*3*	*4*	*5*	*6*	*7*
0	1.00	1.00	1.00	1.00	1.00	1.00
1	0.37	0.61	0.72	0.78	0.82	0.85
2	0.00	0.14	0.37	0.51	0.61	0.67
3		0.00	0.05	0.22	0.37	0.47
4			0.00	0.02	0.14	0.26
5				0.00	0.01	0.08
6					0.00	0.00
7						0.00

Table 2. Membership values for approximate matchings of terms of various lengths.

For the Boolean operations, we use definitions from fuzzy set theory [30]:

$$\begin{aligned} \mathcal{F}^{FB}_{AND}(x, y) &= \min(x, y) \\ \mathcal{F}^{FB}_{OR}(x, y) &= \max(x, y) \\ \mathcal{F}^{FB}_{NOT}(x) &= 1 - x \end{aligned} \qquad (8)$$

Hence, if the document were "President Bill CIinton and Vice President Al Gore met briefly today," then evaluation of the query (Clinton AND Gore) would proceed as follows:

$$\begin{aligned} &\mathcal{Q}^{FB}(D) \\ &= \mathcal{F}^{FB}_{AND}(\mathcal{M}^{FB}(\text{Clinton}, D), \mathcal{M}^{FB}(\text{Gore}, D)) \\ &= \mathcal{F}^{FB}_{AND}(0.85, 1.00) = \min(0.85, 1.00) = 0.85 \end{aligned}$$

The Unix *agrep* utility can be viewed as implementing a restricted form of fuzzy Boolean retrieval [28]. It differs from the model we have presented here in at least two significant ways:

1. Queries can contain AND operators or OR operators, but not both. There is no NOT operator.

2. The user must specify the maximum number of editing operations in advance (the default is 0, *i.e.*, exact matching). The hits are returned in "scan" order (as with standard *grep*), not ranked by "goodness."

Like *grep* however, *agrep* can match regular expressions, a fundamentally different class of patterns than the Boolean queries we are considering.

3.3. Proximity retrieval

Proximity has proven to be another useful concept in information retrieval. As an example, one such query might be "Find all documents where the words 'Clinton' and 'Gore' appear in the same sentence." The precise meaning of "proximity" could, in fact, be defined relative to any logical structure derivable from a document (characters, lines, paragraphs, columns, etc.).

From a notational standpoint, we write a proximity query as [Clinton | Gore], while a more complex query might be ([Clinton | Gore] OR U.S. leaders). That is, a proximity term replaces a simple term in the traditional model. This query also illustrates how proximity might better capture a user's intentions: "Clinton" and

"Gore" in the same sentence is conceptually closer to "U.S. leaders" than "Clinton" and "Gore" in two unrelated sentences far apart in the document.

In effect, proximity is a new form of membership function that takes term-term-document triples and maps them onto a set of values in the interval $[0, 1]$. Rather than treat this as an entirely new query model, we carry over the membership function $\mathcal{M}^B$ and operations $\mathcal{F}^B_{AND}$, $\mathcal{F}^B_{OR}$, and $\mathcal{F}^B_{NOT}$ from the standard Boolean case. To this we add a new membership function $\mathcal{M}^P$ for use in evaluating proximity terms.

We begin by breaking the document D into consecutive, non-overlapping substrings, $D = D_1 D_2 \ldots D_k$, where the partitions are determined by the meaning of "proximity." For example, if the proximity is "in the same sentence," then the document is broken at sentence-ending punctuation (periods, question marks, and exclamation points). Then $\mathcal{M}^P(T_1, T_2, D)$ is defined as:

$$\mathcal{M}^P(T_1, T_2, D) = \begin{cases} 1 & \exists i \text{ such that} \\ & \quad \mathcal{F}^B_{AND}(\mathcal{M}^B(T_1, D_i), \mathcal{M}^B(T_2, D_i)) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Evaluation of a query incorporating proximity terms then proceeds exactly as before. Other forms of proximity are possible, and are handled in a similar way.

3.4. Fuzzy proximity retrieval

The preceding definition of proximity clearly breaks down when the logical structure of the document is not maintained. It is not uncommon for an OCR system to miss line breaks and column and table boundaries, and to delete and insert spaces and punctuation marks (especially periods) throughout the text. All of these error effects can have a large impact on proximity queries.

To incorporate a degree of fuzziness in this case, we allow for the fact that OCR and document analysis errors may have added "noise" to the logical partitioning of the document. (We also continue to allow fuzziness in the basic term membership function as well.) For example, it is possible that the pairing $\mathcal{M}^{FB}(T_1, D_i)$ and $\mathcal{M}^{FB}(T_2, D_j)$, where $i \neq j$, may be used to satisfy the query. We define the membership function as:

$$\mathcal{M}^{FP}(T_1, T_2, D) = \max_{1 \leq i,j \leq k} \left\{ \gamma_{i,j,k} \, \mathcal{F}^{FB}_{AND}(\mathcal{M}^{FB}(T_1, D_i), \mathcal{M}^{FB}(T_2, D_j)) \right\} \tag{10}$$

where $\gamma_{i,j,k} = 1/e^{\beta |i-j| / (k-1-|i-j|)}$ and β is a constant to be determined. This computes the best pairing of individual "hits," weighted by an exponentially decreasing function of the distance between them.

4. Criteria for performance evaluation

The most straightforward measure of success is whether or not a retrieval model returns the appropriate document(s). This is the metric we use here. Another, somewhat different approach would be to consider the actual value assigned to each document, $\mathcal{Q}(D)$, and perform an analysis of the resulting rank order statistics; see [11] for a study of this kind.

It seems inevitable that any retrieval algorithm will miss some documents that truly satisfy the query, and report false "hits" for other documents that do not. The following two criteria are traditionally used to quantify these notions [19]:

Recall	The percentage of true hits that are reported.
Precision	The percentage of reported hits that are in fact true.

It is desirable to have both of these values as close to 1 as possible. There is, however, a well-known trade-off between the two. By insisting on an exact match, the precision can be made 1, but the recall will undoubtedly suffer. On the other hand, if we allow arbitrary amounts of "fuzziness" (and hence return more and more documents), the recall will approach 1, but the precision will fall to 0. For a given database to be searchable, there must exist a point on this trade-off curve where both the recall and the precision are sufficiently high.

Before we define recall and precision in terms of our previous notation, we note that for the fuzzy models, $\mathcal{Q}$ returns a value in the interval $[0, 1]$. Hence, we must be more specific when we speak of a "hit" in these cases. This real value is converted into a 1 (hit) or 0 (miss) based on a pre-determined threshold τ:

$$\mathcal{Q}^*(D) = \begin{cases} 1 & \text{if } \mathcal{Q}(D) \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

The actual value τ can be varied, of course. For the remainder of this paper, we will continue to use the notation $\mathcal{Q}$ knowing that we really mean $\mathcal{Q}^*$ for the fuzzy models.

Let $\Delta = \{D_1, D_2, \ldots, D_n\}$ be the database of original ("clean") documents, and $\Delta^N = \{D^N_1, D^N_2, \ldots, D^N_n\}$ be a database of "noisy" documents. For a given query $\mathcal{Q}$, the set of all "true" hits is:

$$\mathcal{T}(\mathcal{Q}, \Delta) = \{D_i \in \Delta \mid \mathcal{Q}(D_i) = 1\} \tag{12}$$

On the other hand, the set of reported hits that are true is:

$$\mathcal{R}^T(\mathcal{Q}, \Delta, \Delta^N) = \{D^N_i \in \Delta^N \mid \mathcal{Q}(D^N_i) = 1 \text{ and } \mathcal{Q}(D_i) = 1\} \tag{13}$$

while the set of reported hits that are false is:

$$\mathcal{R}^F(\mathcal{Q}, \Delta, \Delta^N) = \{D_i^N \in \Delta^N \mid \mathcal{Q}(D_i^N) = 1 \text{ and } \mathcal{Q}(D_i) = 0\} \quad (14)$$

Then we can define recall as:

$$\text{Recall}(\mathcal{Q}, \Delta, \Delta^N) = \frac{|\mathcal{R}^T(\mathcal{Q}, \Delta, \Delta^N)|}{|\mathcal{T}(\mathcal{Q}, \Delta)|} \quad (15)$$

and precision as:

$$\text{Precision}(\mathcal{Q}, \Delta, \Delta^N) = \frac{|\mathcal{R}^T(\mathcal{Q}, \Delta, \Delta^N)|}{|\mathcal{R}^T(\mathcal{Q}, \Delta, \Delta^N)| + |\mathcal{R}^F(\mathcal{Q}, \Delta, \Delta^N)|} \quad (16)$$

For both of these measures to be 1, the algorithm must report all of the true hits, and only the true hits. The recall (precision) for a collection of queries is computed by averaging the individual recall (precision) values.

To facilitate comparison of the final results, we define "true hit" relative to the sharp models. Note, however, that this may unfairly penalize the fuzzy algorithms by a small amount. For example, if the query term is "company", then the traditional methods will not return a document containing only the term "companies", whereas the fuzzy methods might.

5. Experimental results

5.1. Design of the experiment

In order to study the behavior of the retrieval models just presented, we performed a large-scale experiment involving a total of 59.6 million query evaluations and a database of 1,000 news articles gathered from the Internet.

To simulate the output of an OCR process, we coded in C two different Unix "filters" for generating errors. Using synthetic (as opposed to real) OCR data has several advantages: it allows us a fine degree of control over the damage, and it makes it easier to run the retrieval algorithms on a much wider range of inputs. The simpler of the models we developed injects independent, identically distributed (i.i.d.) random single-character deletion, insertion, and substitution errors. The other model uses a confusion matrix derived from analyzing a large body of OCR output to generate error patterns that are more "realistic-looking."

For a given noise level p, the i.i.d. procedure sequentially scans a document D and for each character c decides, with probability p, whether c is to be altered. If an error is determined to have occurred, the filter then randomly decides whether c is to be deleted or substituted for, or another character is to be inserted just prior to c. The probabilities for these three cases are all equal. For substitution and insertion errors, another uniformly random event selects a character $s \in \Sigma$ which replaces c or is inserted before it, respectively.

In the case of this simple model, the noise level p is roughly equivalent to an OCR error rate computed "after-the-fact" [4]. In Figure 2, we show examples of the output of the i.i.d. filter for levels ranging from "clean" (no errors) to a 40% error rate. For the experiment itself, we varied the p parameter from 0.00 to 0.40 in steps of size 0.02.

Original
```
Chapter I 2 LOOMINGS
Call me Ishmael.  Some years ago -- never mind how long
precisely -- having little or no money in my purse, and
nothing particular to interest me on shore, I thought I
```

10% Noise
```
Chapter I 2 LOOKINGS
Call me Ishmael.  SoGe years ago1 -R- neve^mind h;ow lng
precisel-p havig little or no_ moneyC Hin 8mypurse ad
nothing partigalOarGto nterest me >on shore, I thoought I
```

20% Noise
```
ChLapter I%[2 LOOMINGS
C7allame Ic8mael. Some years@ Zago -- neverUmind ho@ long
precis#elP -- havn# littme ok no.moneZ in my purse, and
nohing&arti[cular.E }interest me $on shore, I though< I
```

30% Noise
```
hapeI2cLWING
Calle Ishmael.K 7 Som yearVago P-M ne-er miQd how? "gOT
prciseVly _-- -havin li(ttle r no EmoneK in wm. prse, .nd
nthinU p)rticul aF toine:est m e onshoor,e, I thought I
```

40% Noise
```
Chhap.tJr I2 LOzOMmINGS
7lwcame?Is-maeH  vome yneJ!rs%a!-'- ne3=r m?Rnd h=owlng
recisely b- h-vi$ lwFoe or no .oneyi my purs5e0) an\
no}hC&n5%patV}cul8r t nter?eGsQE me o zs`ore,' itVhouht
```

Figure 2. Examples of i.i.d. random noise.

The artificial nature of this noise is easy to identify on sight. Another possible approach is to use a confusion matrix to generate error patterns that at least exhibit a proper distribution. In this case, we based our model on OCR results for a long text, the novel *Moby-Dick*. The complete work totals 1,179,194 characters and, when typeset in 10-point Times, requires 322 pages. Because we were scrupulously careful when OCR'ing the clean, first-generation laserprinted copy, the overall accuracy rate was quite high: 99.4%. In other words, the baseline noise level was 0.6%.

The five least- and most-frequently encountered errors are listed in Table 3. Error patterns that did not arise were still assigned a nominal rate of 1×10^{-6}. Since this data only provides one "snapshot" of OCR performance, we applied a linear scale factor to the confusion matrix to obtain other noise levels. (In general, OCR errors do not scale linearly – this simplification was necessary to generate the large amount of data we needed in a reasonable amount of time.) Figure 3 shows examples of the output from the confusion ma-

Least-Frequent		*Most-Frequent*	
Error	*Rate*	*Error*	*Rate*
insert)	0.0001%	1 → l	4.6875%
insert ,	0.0001%	6 → -	4.5455%
insert -	0.0001%	delete -	1.2978%
insert >	0.0001%	O → 0	1.1236%
insert ‘	0.0001%	0 → O	0.9346%

Table 3. Least- and most-frequent OCR errors.

trix filter for levels ranging from “clean” (no errors) to a 40% error rate.

Original

```
Chapter I 2 LOOMINGS
Call me Ishmael.  Some years ago -- never mind how long
precisely -- having little or no money in my purse, and
nothing particular to interest me on shore, I thought I
```

10% Noise

```
Chapte I 2 LOOMINGS
Call me Ishmael.  Some year s ago  ~ ncver mind howlong
precjsely  -- having liltle or~ ao money in my purse,  an~
mothing paticuBar to in terest ne on shore. I th~ught I woul~
```

20% Noise

```
ChapteI I 2 L00MINGS
Call meIsmnel.  Some years ago --  mever  mind how l n g
precisely-- havimg littIe os no moneyI inh my puvse,  nd
noth img particular to int~est me on shore, I thought 1awou lc
```

30% Noise

```
Capte~ I 2 L 00 MINGS
Call m~  IshmaeI. Some yens n go -mever min~ how Iong
pecisely-- having l~ttie ov no money im my pm tse. and
not himg particmlal  to intees  me on shoe, I hought I wou]a
```

40% Noise

```
Chapte l2  L00MI NG S
SaIB m e Isl~ rel.  Someyeas ag~ n%e Ver min~ how Bolg
p~~ciseIy -having littI e ot moaonky  tm mypur se, s n~_n
oth~img pptic uI t  to  ine est ~e ~~l sh>re, I hought I wouBd
```

Figure 3. Examples of confusion matrix noise.

For our document database, we collected 1,000 news articles from the Internet. These covered a variety of subjects (finance, sports, entertainment, etc.). Table 4 gives some simple statistics for the database.

Quantity	*Words*	*Bytes*
Average document	472	2,999
Shortest document	65	364
Longest document	1,400	8,626
Total database	471,998	2,999,137

Table 4. Document database statistics.

Queries were synthesized using an automated procedure. For the Boolean models, a basic template was chosen randomly for each of a fixed percentage of the documents in the database (*e.g.*, (term_1 AND term_2)). The open terms were then filled-in using words from the document (or, in the case of NOT’s, from words present elsewhere in the database but not in the document). We then evaluated the candidate query against the “clean” database under the standard Boolean model, keeping it only if the number of hits fell within a certain range (generally between 1 and 8). A similar procedure was used for the proximity models, except there was only one possible template, [term_1 | term_2]. In this way, the queries in our test set were guaranteed to be “well-behaved,” if perhaps somewhat contrived.

In all, we generated 346 queries for the Boolean tests, and 127 for the proximity. Some examples of the former are: (Solzhenitsyn), (theater AND reviews), and (Baily OR Marcia), while examples of the latter include: [expectations | earnings], [layoff | workers], and [words | language]. There were also many artificial-sounding queries, as might be expected (*e.g.*, ((NOT East-West OR trades) AND Nasdaq)). The average number of documents returned for a Boolean query was 5.9, while for a proximity query it was 4.8.

The complete experiment then consisted of running, for each of the four retrieval models, the appropriate queries against the 1,000 documents at 21 different noise levels under three different sets of noise model/edit cost conditions. For the Boolean models, this yielded $2 \times 346 \times 1,000 \times 21 \times 3 = 43,596,000$ query-to-document evaluations. For the proximity models, the number was $2 \times 127 \times 1,000 \times 21 \times 3 = 16,002,000$. Hence, the experiment involved 59,598,000 evaluations in total.

For each query, we computed the recall and precision. We averaged the results for a particular noise level to obtain graphs of the performance degradation as the OCR damage increased.

5.2. Evaluation

The first two charts, Figures 4 and 5, show the average recall and precision calculated for the sharp and fuzzy Boolean retrieval models for the i.i.d. noise model and two different values of the threshold τ. Here we have assumed that the string edit distance costs are constant, $c_{del} = c_{ins} = c_{sub} = 1$. When $\tau = 0.2$, the fuzzy version exhibits an impressive degree of robustness in terms of recall. By the time the noise level approaches 0.12, the traditional retrieval model is returning fewer than 50% of the true hits, while the fuzzy algorithm captures 95%. For documents that have suffered severe damage (noise levels of 0.36 or greater), the traditional approach misses over 90% of the hits, whereas the fuzzy method still returns over half.

With regards to precision, the fuzzy model is understandably less selective: only about 30% of the hits it returns are “true.” Still, it is generally preferable to return a little too much data than to miss something important. It is also interesting to note that the

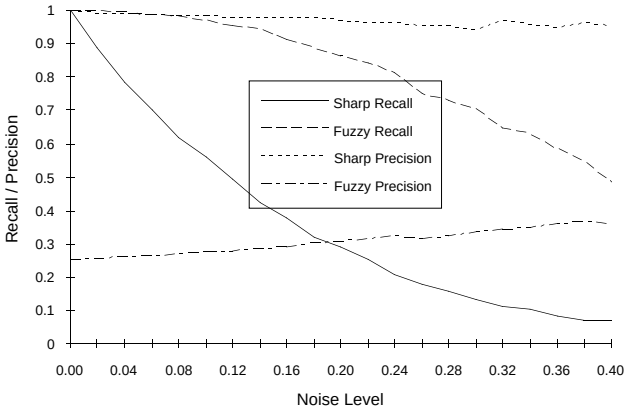

Figure 4. Results for Boolean retrieval under i.i.d. random noise ($\tau = 0.2$).

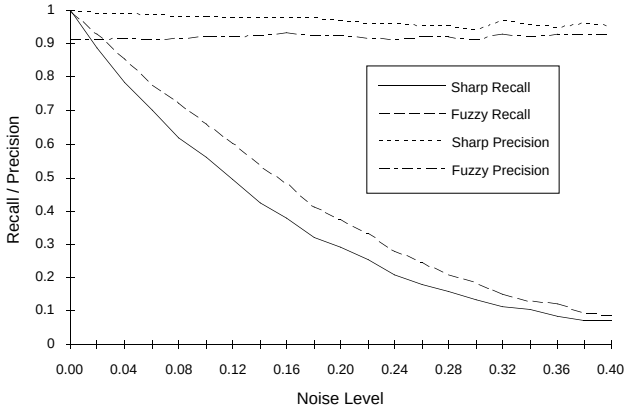

Figure 5. Results for Boolean retrieval under i.i.d. random noise ($\tau = 0.6$).

precision for the sharp model fluctuates, occasionally dropping to near 95%. These cases correspond to documents becoming false hits as a result of the injected noise.

As the threshold τ is increased, the behavior of the fuzzy Boolean retrieval algorithm approaches that of the original. At $\tau = 0.4$ (not shown), it still has an advantage in terms of recall, along with an increase in precision in comparison to $\tau = 0.2$, but by $\tau = 0.6$ (Figure 5) the two methods are less distinguishable.

We present a similar set of recall and precision curves for the two proximity retrieval models in Figure 6. While the fuzzy version has higher recall at $\tau = 0.2$, the improvement is less dramatic. Note that precision increases with the noise level as fewer and fewer documents satisfy the queries. Interaction between the two constants to be set in this case, α and β, may explain this behavior. A more careful "tuning" of these values would probably yield better results.

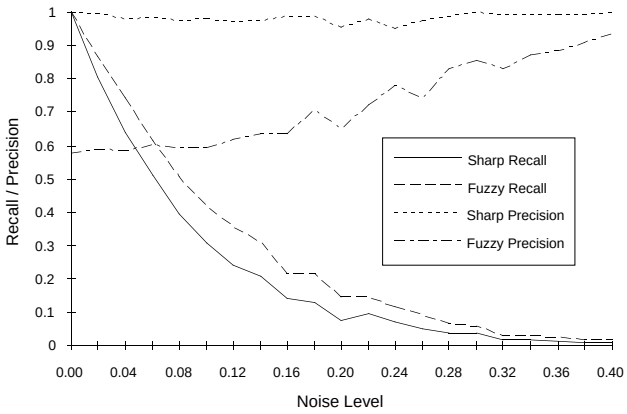

Figure 6. Results for proximity retrieval under i.i.d. random noise ($\tau = 0.2$).

Curves for the confusion matrix noise model are presented in Figures 7 and 8. These two sets of results bear a close resemblance to their i.i.d. counterparts (Figures 4 and 6, respectively). This suggests that the choice of a noise model may be less important than other effects (*e.g.*, the precise degree of damage). The fact that the confusion matrix curves are noticeably "smoother" is almost certainly a sampling anomaly: while every error still has some probability of occurring, the distribution is highly asymmetric so there are effectively fewer error types in this case.

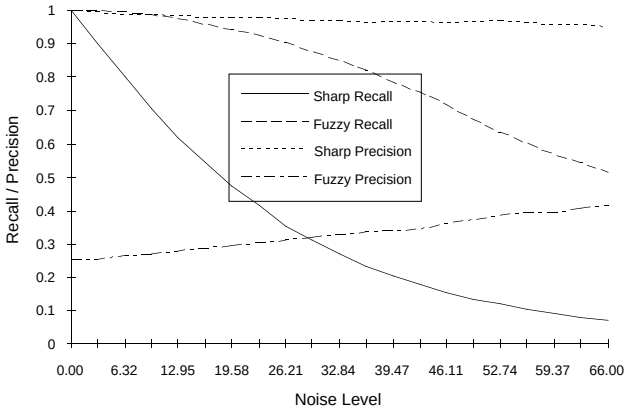

Figure 7. Results for Boolean retrieval under confusion matrix noise ($\tau = 0.2$).

Lastly, we present results in Figures 9 and 10 for when the same confusion matrix is used to generate the noise and determine the string editing costs. This

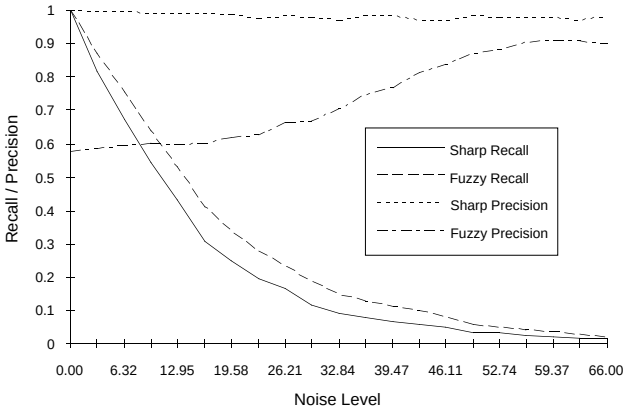

Figure 8. Results for proximity retrieval under confusion matrix noise ($\tau = 0.2$).

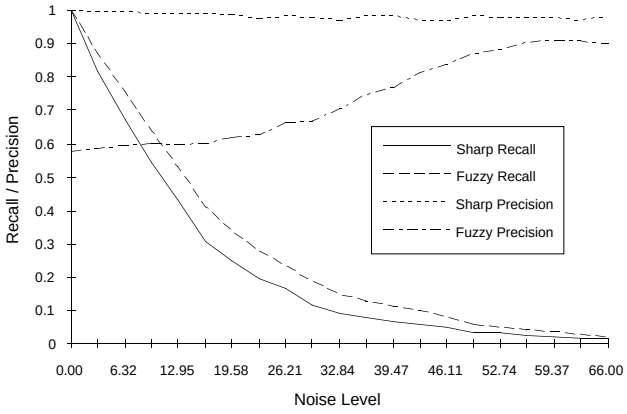

Figure 10. Results for proximity retrieval under confusion matrix noise and edit costs ($\tau = 0.2$).

scenario corresponds to having an accurate model for the OCR noise process. Because this produces a significant change in $\mathcal{E}(T, D)$ and hence the membership function $\mathcal{M}$, we have chosen to use a different threshold ($\tau = 0.8$) for the Boolean case. The curves here are quite dissimilar. Note in particular that recall remains almost constant for the fuzzy Boolean model across the full range of noise levels, from 0% to 40%. Whether this phenomenon would arise in practice is unclear, but is likely to depend on the accuracy of the edit distance computation in "inverting" the effects of the noise source.

Results for the proximity models resemble the previous sets of curves. Evidently the predominate effect in these tests is not the values returned by the term membership function, but the way in which they are combined using Equations 9 and 10.

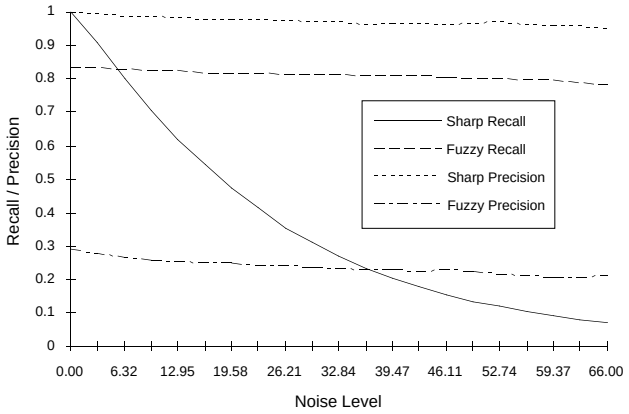

Figure 9. Results for Boolean retrieval under confusion matrix noise and edit costs ($\tau = 0.8$).

6. Conclusions and future research

In this paper, we have examined the effects of simulated OCR errors on Boolean query models for information retrieval. While the "sharp" methods are sensitive to such noise, new versions based on approximate string matching and fuzzy logic seem to be more robust.

Although the fuzzy algorithms we presented are promising, much work remains to be done in the area of efficiency. There exist well-known data structures and algorithms for indexing documents and searching them quickly [6]. Most of these are lexicon-based (*i.e.*, the terms are pre-defined). For the fuzzy models we have described, the existence of such lexicons seems to be an open problem.

We note here briefly that Aref and Barbará discuss a heuristic approach based on a trie data structure that seems to work well for a similar problem involving electronic ink (handwriting) [1]. Also, Myers gives a sub-linear time algorithm for the special case where the number of errors is bounded in advance by a constant [13]. For the string matching portion of the computation, it is possible to use a parallel VLSI architecture to speed the search [10].

Another interesting question is whether it is possible to estimate automatically the noise level of a particular document. If this were the case, we could use such information to adapt the weights α and β dynamically in the fuzzy matching algorithms. This would undoubtedly lead to better recall/precision performance.

We also note that existing studies of OCR noise and its impact on information retrieval (including our own) have examined only one class of errors that might

arise: character-oriented. Other types of DIA errors are clearly relevant, including the merging of columns, failure to detect table cell boundaries, large-scale deletions of text and insertions of "gibberish" (*i.e.*, burst errors), etc. Similar recognition mistakes are likely to arise even in the analysis of purely electronic documents. From the standpoint of information retrieval, such effects warrant further study.

Finally, we note that most popular general-purpose World Wide Web search engines do not currently support fuzzy matching [3, 5, 7, 12, 16, 26, 29]. In each case, the query "Panasonic" turned up many hits, while the query "Panasoanic" failed to find any. It seems likely that this application area could benefit from the inclusion of fuzzy retrieval methods.

7. Acknowledgements

The author thanks Drs. Jiangying Zhou and Jeffrey Zhou of MITL for their assistance and many helpful discussions. The trademarks mentioned in this paper are the properties of their respective companies.

References

[1] W. G. Aref and D. Barbará. An indexing scheme for cursive handwritten databases. Technical Report 140-95, Matsushita Information Technology Laboratory, Princeton, NJ, October 1995.

[2] W. B. Croft, S. M. Harding, K. Taghva, and J. Borsack. An evaluation of information retrieval accuracy with simulated OCR output. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 115–126, Las Vegas, NV, April 1994.

[3] DejaNews. http://www.dejanews.com/.

[4] J. Esakov, D. P. Lopresti, J. S. Sandberg, and J. Zhou. Issues in automatic OCR error classification. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 401–412, Las Vegas, NV, April 1994.

[5] Excite. http://www.excite.com/.

[6] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[7] InfoSeek. http://www2.infoseek.com/.

[8] D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, NV, April 1995.

[9] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[10] R. J. Lipton and D. P. Lopresti. A systolic array for rapid string comparison. In H. Fuchs, editor, *Proceedings of the 1985 Chapel Hill Conference on VLSI*, pages 363–376. Computer Science Press, 1985.

[11] D. Lopresti and J. Zhou. Retrieval strategies for noisy text. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, Las Vegas, NV, April 1996. To appear.

[12] Lycos. http://www.lycos.com/.

[13] E. W. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12(4-5):345–374, 1994.

[14] A. Myka and U. Güntzer. Fuzzy full-text searches in OCR databases. In *Advances in Digital Libraries*, chapter 7. Springer-Verlag, New York, NY, 1995.

[15] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino-acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[16] Open Text. http://www.opentext.com/.

[17] C. Pearce. Dynamic hypertext links for highly degraded data in TELLTALE. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 89–106, Las Vegas, NV, April 1995.

[18] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fourth annual test of OCR accuracy. In *Annual Report of UNLV Information Science Research Institute*, pages 11–49, Las Vegas, NV, 1995.

[19] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[20] K. Taghva, J. Borsack, and A. Condit. Effects of OCR errors on ranking and feedback using the vector space model. Technical Report 94-06, UNLV ISRI, Las Vegas, NV, August 1994.

[21] K. Taghva, J. Borsack, and A. Condit. Results of applying probabilistic IR to OCR text. In *Proceedings of the Conference on Research and Development in Information Retrieval*, pages 202–211, Dublin, Ireland, July 1994.

[22] K. Taghva, J. Borsack, A. Condit, and S. Erva. The effects of noisy data on text retrieval. *Journal of the American Society for Information Science*, 45(1):50–58, January 1994.

[23] K. Taghva, J. Borsack, A. Condit, and P. Inaparthy. Effects of OCR errors on short documents. In *Annual Report of UNLV Information Science Research Institute*, pages 99–105, Las Vegas, NV, 1995.

[24] K. Tsuda, S. Senda, M. Minoh, and K. Ikeda. Clustering OCR-ed texts for browsing document image database. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 171–174, Montréal, Canada, August 1995.

[25] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

[26] WebCrawler. http://www.webcrawler.com/.

[27] L. Wiedenhöfer, H.-G. Hein, and A. Dengel. Post-processing of OCR results for automatic indexing. In *Proceedings of the Third International Conference on*

Document Analysis and Recognition, pages 592–596, Montréal, Canada, August 1995.

[28] S. Wu and U. Manber. Fast text searching with errors. Technical Report 91-11, University of Arizona, Tucson, AZ, June 1991.

[29] Yahoo. http://www.yahoo.com/.

[30] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–352, 1965.