Retrieval Strategies for Noisy Text

Daniel Lopresti and Jiangying Zhou
[dpl,jz]@mitl.research.panasonic.com
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way
Princeton, NJ 08540

Abstract

In this paper, we examine the effects of various kinds of simulated OCR noise on the performance of several popular information retrieval models. We also introduce new versions of some of the models by combining two classic paradigms for dealing with imprecise data: approximate string matching and fuzzy logic. Based on a rank-order analysis of experimental data, we show that the new fuzzy retrieval models appear to be generally more robust than their traditional counterparts.

1 Introduction

When studying information retrieval models and algorithms, it is often convenient to assume that the contents of the database are, in a sense, perfect: all of the text belongs, nothing is missing, and there are few if any errors. With the growing use of fully-automated OCR and document analysis systems, however, it is becoming increasingly evident that significant levels of "noise" may be present in the data. Even state-of-the-art OCR technology produces garbled output when confronted by unanticipated document formatting (*e.g.*, shaded backgrounds, unusual fonts, white-on-black text).

Moreover, text taken directly from electronic sources (*e.g.*, Usenet postings) may also contain spurious information (from the standpoint of retrieval), as well as typographic errors. For example, documents may be encountered in any number of formats that contain a large amount of "metadata" (*i.e.*, formatting instructions), including HTML, RTF, PostScript, LaTeX, etc. Totally "clean" documents are likely to be the exception rather than the rule.

Figure 1 illustrates the relationship between various classes of document analysis errors and several popular models for information retrieval. While some schemes are relatively robust in the presence of certain types of noise, in other cases the impact may be much more dramatic.

In this paper, we examine the effects of various kinds of simulated OCR noise on the performance of several popular information retrieval models. A consistent notation is presented and used throughout the discussion, making it possible to compare the various components of the models directly, and to judge their similarities and differences on a common framework.

We also introduce new versions of some of the retrieval models by combining two classic paradigms for dealing with imprecise data: approximate string matching and

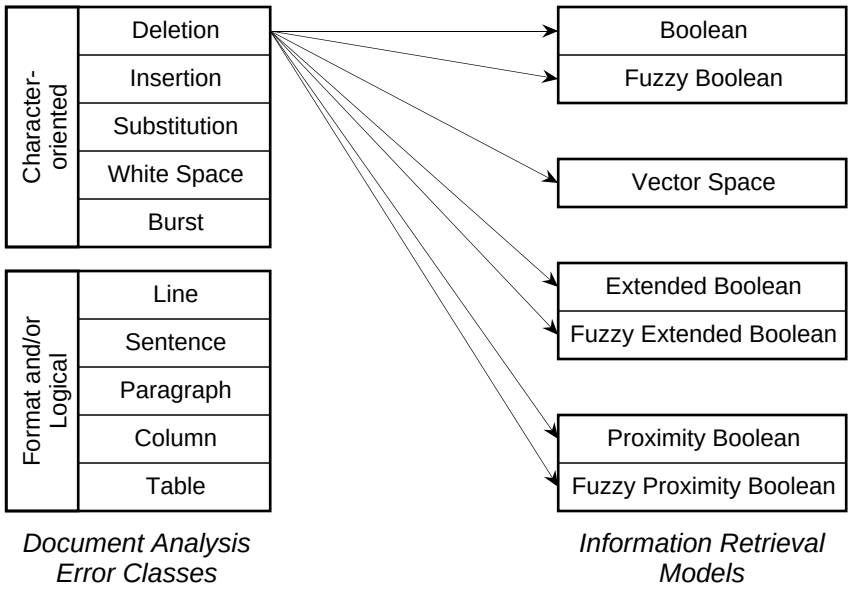

Figure 1: Document analysis error classes and information retrieval models.

fuzzy logic. Based on a rank-order analysis of experimental data, we show that certain schemes are more robust than others, and that the new fuzzy models are generally more robust than their traditional counterparts.

2 Related Work

Researchers have done studies to identify the effects of OCR errors with respect to different models of information systems. These studies have suggested that traditional information retrieval techniques have some resilience to degradation in the text, particularly with long documents.

For example, Taghva, Borsack, Condit, and Gilbreth [1, 2] considered the effects of several commercial OCR packages on retrieval accuracy of a vector space model *SMART* and a probability model *INQUERY*. They concluded that OCR systems with high recognition rates have minimal effect on retrieval performance for long documents. Recently, Mittendorf, Schäuble and Sheridan [3] provided a theoretical explanation of this phenomena for a particular retrieval method. Interestingly, in Taghva *et al.*'s papers, the authors stated that although the average precision and recall are not significantly affected by common OCR errors, individual queries can be greatly affected sometimes. Also, when different weighting schemes are applied, the ranking observed between the OCR version and a "corrected" version are affected. Experiment results on SMART system also show that the average precision and recall are not affected for short documents either [4]. In addition, the performance difference on an OCR database and on a post-processed version is not significant for short documents. In a paper by Taghva, Borsack, Condit, and Erva [5] an evaluation on *BASISplus* system which is based on Boolean model is reported. The result shows that the precision is not affected on-average by OCR damage.

On the other hand, Croft, Harding, Taghva, and Borsack [6] used a simulated

OCR model to study the effect of noise on a probabilistic retrieval model. The result shows that noise has an impact on short documents. Recently, Tsuda, Senda, Minoh, and Ikeda [7] studied how simulated single character substitutions may affect a vector space clustering model. Their conclusion is that the system may be able to tolerate up to 15.6% character damage. A similar conclusion was reached by Ittner, Lewis, and Ahn [8] in a paper which shows how the vector space method for document clustering has sustained badly damaged faxed text.

Several papers also suggested ways to cope with OCR noise in the design of a document retrieval system. For example, Pearce suggested using n-grams for robustness [9] in building a hypertext links using degraded OCR output. Several fuzzy matching methods for reducing the impact of OCR noise were proposed by Myka and Güntzer [10]. Mittendorf, Schäuble and Sheridan [3] proposed a probabilistic weighting approach to indexing short OCR text. In a paper by Wiedenhöfer, Hein, and Dengel [11] the authors present an indexing component whose input are character hypotheses lattices which are post-processed by a generate-and-test component feeding a morphology, a rule based substitution system, and a trigram correction component with word candidates.

3 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 to compare directly the various model components, and to judge their similarities and differences on a common framework.

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 given term is expressed in a particular document. In the extreme, $\mathcal{M}(T, D) = 1$ when the term's presence is strong, and $\mathcal{M}(T, D) = 0$ when the term is completely absent. For some models, the membership function is defined relative to term-document-database triples, $\mathcal{M}(T, D, \Delta)$. Here the interpretation reflects the expression of the term in the document relative to its expression in the other documents in the database.

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:

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

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

Lastly, a *ranking* $\mathcal{R}$ is a mapping between the documents in a database Δ

and integers in the range $[1, |\Delta|]$ such that $\mathcal{R}(D_i) = r$ if and only if $|\{D_j \in \Delta \mid \mathcal{R}(D_j) < r\}| = r - 1$. That is, a document is assigned rank r if and only if $r - 1$ documents in the database are assigned some higher rank. Thus, for example, $\{1, 4, 1, 3, 5\}$ is a legal ranking for the document database $\{D_1, D_2, D_3, D_4, D_5\}$, whereas $\{1, 4, 1, 4, 5\}$ is not. The evaluation of a query induces a ranking on the documents in a database in an obvious way.

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:

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

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

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 operators.

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".

Note that the rankings produced by the simple Boolean model are somewhat degenerate (but still valid from a mathematical standpoint). All of the documents satisfying the query, say there are m of them, are assigned rank 1. The rest of the documents in the database are assigned rank $m + 1$.

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 appears 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. 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* [12], 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 [13, 14]. 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 $\delta_{i,j}$ to be the distance between the first i characters of T and the first j characters of D. The initial conditions are:

$$\begin{aligned} \delta_{0,0} &= 0 \\ \delta_{i,0} &= \delta_{i-1,0} + c_{del}(t_i) \\ \delta_{0,j} &= \delta_{0,j-1} + c_{ins}(d_j) \end{aligned} \quad (3)$$

and the main dynamic programming recur-

rence is:

$$\delta_{i,j} = \min \begin{cases} \delta_{i-1,j} & + c_{del}(t_i) \\ \delta_{i,j-1} & + c_{ins}(d_j) \\ \delta_{i-1,j-1} & + c_{sub}(t_i, d_j) \end{cases} \quad (4)$$

When Equation 4 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 is sometimes called "word spotting"). We make the initial edit distance 0 along the entire length of the document (allowing a match to start anywhere), and search the final row of the edit distance table for the smallest value (allowing a match to end anywhere). The initial conditions become:

$$\begin{aligned} \delta_{0,0} &= 0 \\ \delta_{i,0} &= \delta_{i-1,0} + c_{del}(t_i) \\ \delta_{0,j} &= 0 \end{aligned} \quad (5)$$

The inner-loop recurrence (*i.e.*, Equation 4) remains the same.

Figure 2 shows the results of this computation for a simple example where we have assumed that $c_{del} = c_{ins} = c_{sub} = 1$. 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").

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 $\mathcal{E}(T, D) = \min(\mathcal{E}(T, D, i) \mid 0 \leq i \leq n)$ corresponds to the best match(es). Under the 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 with range $[0, 1]$ for this model through the use of an exponential decay:

$$\mathcal{M}^{FB}(T, D) = \frac{1}{e^{\alpha \mathcal{E}(T,D)/(m - \mathcal{E}(T,D))}} \quad (6)$$

where α is a constant.

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.

For the Boolean operations, we use definitions from fuzzy set theory [15]:

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

Hence, if the document were "The q~ick brown tox jurnps ovcr the lazy dog." then evaluation of the query (fox AND dog) would proceed as follows:

$$\begin{aligned} \mathcal{Q}^{FB}(D) &= \mathcal{F}^{FB}_{AND}(\mathcal{M}^{FB}(\text{fox}, D), \mathcal{M}^{FB}(\text{dog}, D)) \\ &= \mathcal{F}^{FB}_{AND}(0.607, 1.000) \\ &= \min(0.607, 1.000) \\ &= 0.607 \end{aligned}$$

The Unix *agrep* utility can be viewed as implementing a restricted form of fuzzy Boolean retrieval [16]. It differs from the model we have presented 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 *a priori* (the default is 0, *i.e.*, exact matching). The hits are returned in "scan" order (as with standard *grep*), not ranked by "goodness."

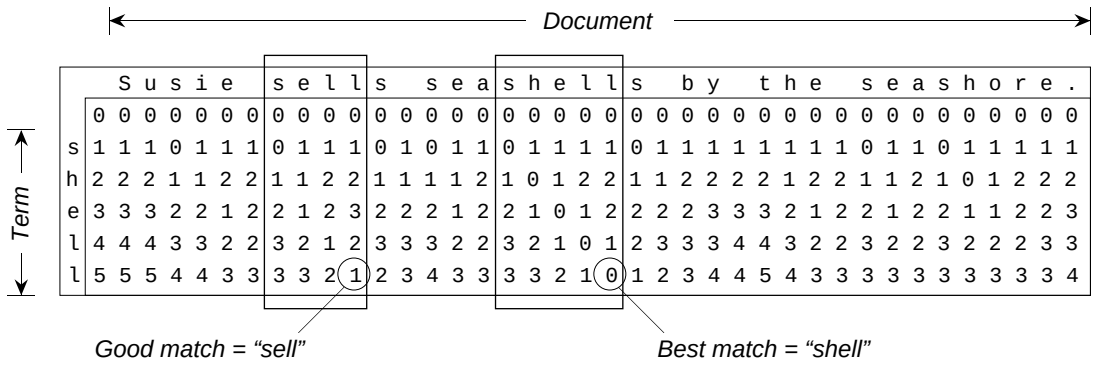

Figure 2: An example of edit distance "word spotting".

Edit Distance $\mathcal{E}(T,D)$	*Query Term Length (m)* 2	3	4	5	6	7	8
0	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1	0.368	0.607	0.717	0.779	0.819	0.846	0.867
2	0.000	0.135	0.368	0.513	0.607	0.670	0.717
3		0.000	0.050	0.223	0.368	0.472	0.549
4			0.000	0.018	0.135	0.264	0.368
5				0.000	0.007	0.082	0.189
6					0.000	0.002	0.050
7						0.000	0.001
8							0.000

Table 2: Membership values for approximate matchings of various length terms.

Like *grep*, *agrep* can match regular expressions, a fundamentally different class of patterns than Boolean queries. Despite their power, however, regular expressions are not commonly supported in query languages for document databases.

3.3 Vector Space Retrieval

Another important information retrieval model is the Vector Space model originally proposed by Salton and Lesk [17]. Here, the similarity between a query and a document is derived from a statistical notion. Both document sets (databases) and query terms are represented as vectors of real-valued weights:

$$< w_1^i, w_2^i, ..., w_t^i >$$

where w_k^i is the weight assigned to the k^{th} indexing term in the i^{th} document. An indexing term may be a word, word-stem, phrase, character n-gram, or other linguistic entity.

Several techniques have been used to compute these weights, the most common being $tf \times idf$ which is based on the frequency of a term in a single document (tf) and its frequency in the entire collection (idf):

$$w_k^i = tf_k^i \times \log\left[\frac{n}{df_k}\right] = tf_k^i \times idf_k \qquad (7)$$

$$\text{where} \quad idf_k = \log\left[\frac{n}{df_k}\right]$$

tf_k^i is the term frequency in the i^{th} document for the k^{th} indexing term, idf_k is the

inverse document frequency of the k^{th} term, and n is the number of terms in the i^{th} document.

These weights represent the relative "importance" of a term in a given document/query, and can be equated precisely with the notion of a membership function:

$$\mathcal{M}^V(T_k, D_i, \Delta) = w_k^i \qquad (8)$$

To compute the vector representation of a document, the following steps are usually followed:

1. Identify individual words in the document.
2. Remove high frequency functional words (stop-words) according to a predefined list.
3. Reduce each word to a word-stem using a stemming algorithm.
4. For each remaining word-stem (*i.e.*, term), assign a weighting factor.

An evaluation of query $\mathcal{Q}$ running on document D can be calculated using any of several similarity functions, the most common being the cosine of the query vector $\mathbf{u}$ and the document vector $\mathbf{w}$:

$$\mathcal{Q}^V(D) = \frac{\sum_k w_k u_k}{\sqrt{\sum_k w_k^2 \sum_k u_k^2}} \qquad (9)$$

Note that according to the above scheme, long documents are indexed by a larger number of terms, and thus have a better chance of being retrieved than shorter documents. A remedy for this bias is to normalize the vector. Various such schemes have been proposed in the literature. A typical approach [18], which we have adopted for our experiments, is:

$$w_k^i = ntf_k^i \times nidf_k \qquad (10)$$

where $ntf_k^i = \frac{tf_k^i}{\max_k tf_k^i}$, $nidf_k = \frac{\log\left[\frac{n}{df_k}\right]}{\log(n)}$

3.4 Extended Boolean Retrieval

The standard Boolean model is unable to distinguish between degrees of usage: either a term is present in a document or it is not. To bridge this gap, Salton, *et al.* have proposed using weighted Boolean operators, known as the Extended Boolean (or p-norm) model [19]. In this model, the ranking factor depends not only on the number of keywords in common between the query and the document, but also on weights assigned to index terms and on a p-value associated with each query operator.

In terms of our previous formulations, the membership function is the same as for the Vector Space model:

$$\mathcal{M}^{EB}(T, D, \Delta) = \mathcal{M}^V(T, D, \Delta) \qquad (11)$$

The Boolean operators in this case are defined by:

$$\mathcal{F}^{EB}_{AND}(x, y) = 1 - \left[\frac{(1-x)^p + (1-y)^p}{2}\right]^{1/p}$$

$$\mathcal{F}^{EB}_{OR}(x, y) = \left[\frac{x^p + y^p}{2}\right]^{1/p}$$

$$\mathcal{F}^{EB}_{NOT}(x) = 1 - x$$

where p is a parameter to be determined. Setting p to be infinity leads to the standard Boolean operators. Setting p equal to 1 eliminates the distinction between AND and OR. For the purposes of this paper, we assume that $p = 2$.

3.5 Fuzzy Extended Boolean Retrieval

Our approach in this case is a significant departure from the traditional models. Unlike the Vector Space and Extended Boolean methods, we do not decompose documents and query text into stemmed terms in advance. Instead, we use the results of the approximate string matching computation directly. Consider the final row of Figure 2; recall that these are the values $\mathcal{E}(T, D, i)$.

If the query term (or something like it) occurs frequently in the document, there will be many small values along this row.

The general strategy in this case is to form a weighted sum of edit distance values (biased towards perfect and near matches), and then normalize this by dividing by the length of the document. This membership function is:

$$\mathcal{M}^{FEB}(T, D) = \frac{1}{n} \sum_{i=0}^{n} \frac{1}{e^{\alpha \mathcal{E}(T,D,i)/(m - \mathcal{E}(T,D,i))}} \quad (12)$$

where, again, α is a constant that controls the rate of exponential decay.

Unlike some of the other membership functions we have discussed, Equation 12 is unlikely to return values close to 1.0 except for pathological cases (such as a document and a query composed solely of the same character). This in itself is not a problem, however, as the rankings it induces are appropriate, as will be shown in Section 5.

The Boolean operators are the same as for the extended Boolean model, namely:

$$\begin{aligned} \mathcal{F}^{FEB}_{AND}(x, y) &= \mathcal{F}^{EB}_{AND}(x, y) \\ \mathcal{F}^{FEB}_{OR}(x, y) &= \mathcal{F}^{EB}_{OR}(x, y) \\ \mathcal{F}^{FEB}_{NOT}(x) &= \mathcal{F}^{EB}_{NOT}(x) \end{aligned}$$

In addition to its use of fuzziness, this approach has another advantage over the more traditional models in that any string of characters (including spaces and punctuation) is allowable as a query term, not just stemmed words. On the other hand, the lack of a predefined dictionary may be a disadvantage: it is not immediately obvious how to weigh the occurrences of a particular term relative to all other terms in the document (or database). This issue is an open question.

3.6 Proximity Boolean Retrieval

Proximity is a useful concept that, so far, has not received enough attention in studies of OCR noise and its impact on 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 notional standpoint, we write a simple proximity query as [Clinton | Gore], while more complex query might be ([Clinton | Gore] OR U.S. leaders). That is, a proximity query replaces a single term in the traditional model. This query also illustrates how proximity better captures meaning when two terms appear near one another: "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 end, we add a new membership function $\mathcal{M}^{PB}$ 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). The membership function is defined as:

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

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

3.7 Fuzzy Proximity Boolean 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 $\mathcal{M}^{FB}(T_1, D_i)$ and $\mathcal{M}^{FB}(T_2, D_j)$, where $i \neq j$, may be used to satisfy the query. Let

$$\mathcal{G}(T_1, T_2, D, i, j) = \mathcal{F}^{FB}_{AND}(\mathcal{M}^{FB}(T_1, D_i), \mathcal{M}^{FB}(T_2, D_j))$$

We define the membership function as:

$$\mathcal{M}^{FPB}(T_1, T_2, D) = \max_{1 \leq i,j \leq k} \{\gamma(i,j,k)\mathcal{G}(T_1, T_2, D, i, j)\} \quad (13)$$

where

$$\gamma(i,j,k) = \frac{1}{e^{\beta|i-j|/(k-1-|i-j|)}}$$

is called the weight factor and β is a constant. This computes the best pairing of individual hits, weighted by an exponentially decreasing function of the distance between them.

4 Performance Evaluation

In this section, we study the performance of the various retrieval models for data exhibiting simulated OCR noise. We begin by discussing our noise models. Then we present our evaluation criterion, which is the correlation between the document rankings produced for noisy data relative to those produced for clean data for a given query. After this, we describe our test database and the techniques we used to generate sample queries. Finally, we give the results of running several large-scale experiments.

4.1 Noise Models

We used a noise generator for simulating errors in the document text. Using synthetic noise model has the advantage of being able to control the distribution of noise and subsequent errors as well as to corrupt the same underlying data in a variety ways. We first consider a simple case – that noise to the database is made of random single character insertions, deletions or substitutions. A damage rate p controls the percentage of character errors in the text. In our model, insertion, deletion, and substitution occur in equal probability.

Uniformly distributed character errors, however, are not the type of noise typically seen in a text document. Ideally, one would want to measure the effect of noise under a real world noise model. Unfortunately, a comprehensive and realistic noise model for documents has not yet developed. There have been work on modeling realistic OCR errors by computer simulation [20]. In this paper, in addition to the simple noise model, we present a preliminary study on the effect of two other types of simulated noise model, namely, the *simple-burst model* and the *confusion-matrix model.*

In the simple-burst model, in addition to the single character deletion, inser-

tion, or substitution described in the simple model, burst errors are added randomly over the text according to a pre-specified burst damage rate. The size of burst is determined by Gaussian processes. In the test, we fixed burst damage rate to be 0.5%. Two Gaussian processes $N(3.0, 1.0)$ and $N(30.0, 1.0)$ were used for choosing burst size. The confusion-matrix model generates character errors in proportion to an OCR-type error distribution specified by a confusion matrix. The confusion matrix was generated by running our error classification algorithm [21] on a large quantity of text produced by a commercial OCR package.

4.2 Performance Measure

The issue we are to examine is how retrieval schemes may be affected by damage to the database. Many papers reported data showing the impact of noise in terms of average recall and precision accuracies [5, 6]. Measuring precision and recall, however, requires relevance judgment for all the queries. For some retrieval models (*e.g.*, vector space) this is a rather subjective attribute. In this paper, we use a correlation coefficient analysis to measure the effect of noise on retrieval performance.

Let $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ be two sequences of real number. We define the correlation coefficient between the two sequences as:

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2\right]^{1/2}} \quad (14)$$

Where

$$\bar{x} = \sum_{1}^{n} x_i \qquad \text{and} \qquad \bar{y} = \sum_{1}^{n} y_i$$

In our test, the correlation is calculated between rankings induced by a query on a database and its noisy counterpart.

The correlation coefficient can be viewed as a measure of similarity between the two rankings. As an example, in Table 3 we show that given a database of ten documents and a query, how the ranking order, as computed by the Extended Fuzzy Boolean method, is changed under three different levels of simulated noise (simple model) and how the variation affects the correlation coefficient.

The correlation coefficient in table 3 is computed on the entire database. In other words, every ranking variation counts. From document retrieval point of view, however, variations at the low end of the rankings account for a diminutive role to the retrieval effectiveness. Therefore, it is desirable to compute the correlation coefficient only at the high end of the document rankings.

Suppose $\mathcal{R}(D_1), \mathcal{R}(D_2), \ldots, \mathcal{R}(D_N)$ is the ranking produced by a retrieval model on documents in a database Δ. Let $\mathcal{R}'(D_1), \mathcal{R}'(D_2), \ldots, \mathcal{R}'(D_N)$ be the ranking assigned by the same retrieval model running on a somehow damaged Δ. Further, suppose we are only interested in the rank change at, say, the top $\eta \times 100$ $(0 < \eta \leq 1)$ percent documents in the database. let $N' = \lceil \eta \times N \rceil$ be the the least integral value greater than or equal to $\eta \times N$. Let

$$\Lambda(i) = \begin{cases} i & i \leq N' \\ N'+1 & i > N' \end{cases}$$

The correlation coefficient is then computed between the two sequences $\Lambda(\mathcal{R}(D_i))$ and $\Lambda(\mathcal{R}'(D_i))$.

4.3 Test Data

For testing, we collected 1000 news articles from the Internet, whose topics range from agriculture, to insurance, to transportation.

	Rank			
Doc.	*0% Noi.*	*2% Noi.*	*20% Noi.*	*40% Noi.*
d_1	1	1	3	7
d_2	2	3	5	5
d_3	3	5	2	3
d_4	4	4	4	1
d_5	5	6	7	4
d_6	6	2	1	2
d_7	7	7	6	9
d_8	8	8	8	6
d_9	9	9	9	10
d_{10}	10	10	10	8
ρ	1.0	0.87	0.73	0.49

Table 3: Rank order variations and correlation coefficients.

Table 4 gives some simple statistics for the dataset.

Quantity	*Words*	*Bytes*
Shortest document	42	271
Longest document	656	3,830
Average document	323	2,005
Total database	322,922	2,004,762

Table 4: Database statistics for 1,000 test documents.

Our queries were generated using an automated procedure. In general, we synthesized representative queries for a fixed percentage of the documents in the database. For the Boolean and Extended Boolean models, a basic template was chosen randomly for each designated document:

- (term)
- (term_1 AND term_2)
- (term_1 OR term_2)

The term(s) greater than a pre-determined length were then selected randomly from non-stop-words in the document. For an AND query, either or both of the operands could be complemented, in which case the associated term(s) where chosen from a list of all non-stop-words present somewhere in the database, but not in the document in question. We then evaluated the query against the entire database using the standard Boolean model, keeping only those that hit at least four documents and at most eight. In the event that a query did not satisfy all of these constraints, a new random query of the same type was generated and the process repeated. In this way, the queries in our test set were guaranteed to be "well-behaved," if perhaps somewhat contrived.

A similar procedure was used for the Proximity models. Here, though, we only synthesized queries of the form [term_1 | term_2].

The Vector Space model required a slightly different approach. As before, we generated queries corresponding to specific documents. A random "snippet" of text in average 200 characters long was chosen. Since in this case the question of whether a given query hits other documents is more ambiguous, we did not apply a minimum/maximum hits criterion as before.

Lastly, the query test sets were reviewed by a human proof-reader to catch any anomalies that might have been missed

during the generation process. In all, 400 queries were created for each group of related retrieval models. Some examples of the Boolean and Proximity queries we used are listed in Table 5. Among the 400 Boolean queries, 146 were single term queries, 136 were OR queries, and 118 were AND queries.

Boolean	*Proximity*
(gains AND Corp)	[business | deficit]
(Civil AND NOT recover)	[Ford | Motor]
(desired OR Financing)	[caused | loss]

Table 5: Examples of queries used.

5 Experiments

The experiments run as follow: an increasing level of noise is introduced to the database. At each noise level, documents are ranked against a given query by the retrieval models respectively. The correlation coefficients are then calculated between the ranks calculated on the original data and on their corresponding noised data. Finally, the coefficient is averaged over the entire query set.

The first chart (Figures 3) shows the average correlation coefficient calculated at top 10% ranking order for four Boolean retrieval models as a function of increasing level of added noise. Figures 4 shows the same calculation for the Vector Space retrieval model. Likewise, Figures 5 shows the coefficient curves for the two Proximity retrieval models.

Both Boolean and Vector Space model results show a somewhat linear decreasing correlation between the ranking observed on original data and the ranking on noisy data as the degree of noise level increases. The proximity Boolean model has a greater degree of degradation in rank correlation. We see that among retrieval models, those that use exact matching, namely, Sharp Boolean, Extended Boolean, Vector Space, and Proximity Boolean have lower degree of correlation than models that employ approximate search. A higher correlation could mean the retrieval model is more resilient to data damage.

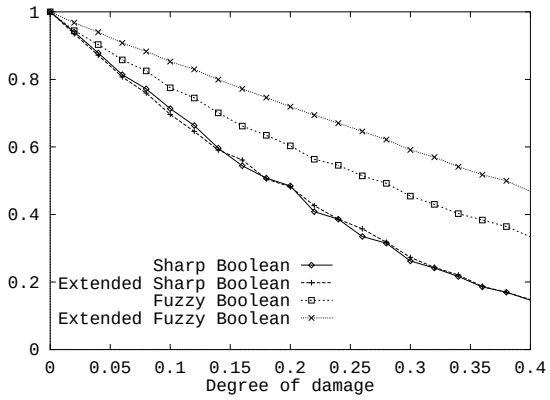

Figure 3: Ranking sensitivity for the Boolean retrieval models.

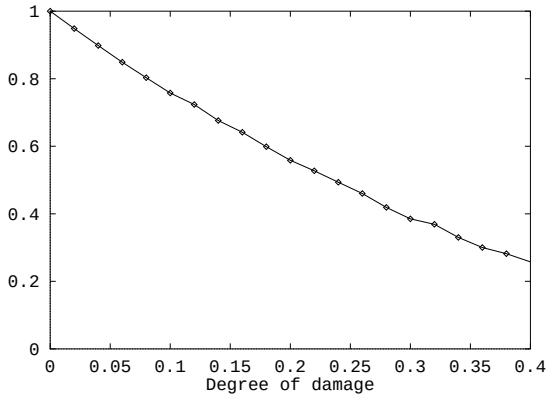

Figure 4: Ranking sensitivities for the Vector Space retrieval model.

Figure 6 compares the correlation coefficient curves of the simple noise model with the two other noise models (Only 200 queries were used in these set of ex-

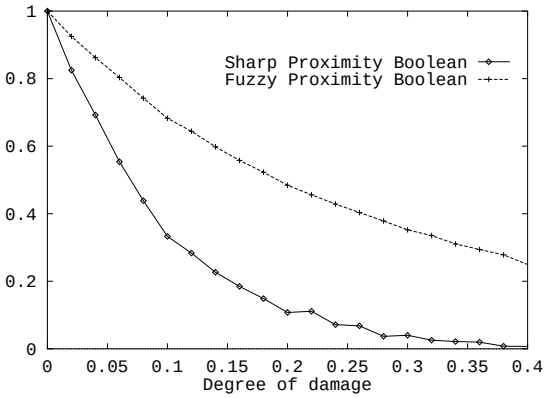

Figure 5: Ranking sensitivities for the Proximity retrieval models.

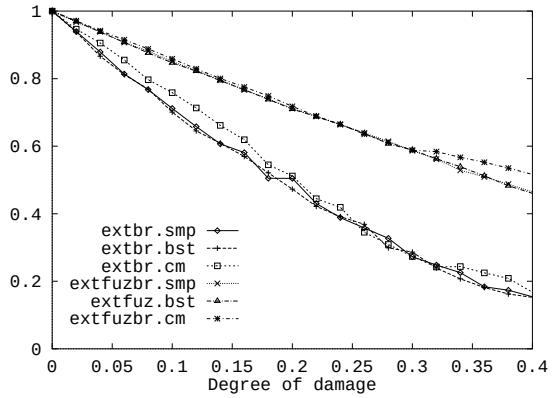

Figure 6: Ranking sensitivities for the extended Boolean models under different noise models.

periments). Curves "extbr.bst" and "extfuzbr.bst" in Figure 6 are the correlation curves of Extended Boolean and Extended Fuzzy Boolean models generated by introducing a burst noise. Curves "extbr.cm" and "extfuzbr.cm" in Figure 6 show the correlation curves where character errors were generated using a confusion matrix. Our preliminary experimental results indicated that the characteristic of rank correlation decay of these three simulated noise types is quite similar for the same retrieval models.

6 Conclusions

Retrieval models based on pre-calculated index structure are relatively fast. Many sophisticated methods have been proposed for efficient access of indices. There are well-known efficient data structures and algorithms for breaking documents into indices (based on a dictionary) [22]. In terms of response time, such models seems to be a choice when one expects little noise in the database.

On the other hand, noise in a database can increase the volume of index dramatically. Figure 7 shows a rapid growth of the index set for vector space model in the test as the noise level increases.

Retrieval models based on sub-string matching have the advantage that they can search for terms in the text which only partially match the input query. When a database contains a high degree of noise, it is desirable to have a retrieval system which torelates errors in text. In addition, substring based retrieval systems allow a user to specify any type of query text (*e.g.*, a phrase, a substring of a word).

Unfortunately, for the sub-string-based models we described, the issue of efficiency is an open problem. We intend to investigate the speed-up of the string matching-based retrieval models in our future work.

Acknowledgments

The authors would like to thank Dr. Jeffrey Zhou for his assistance in generating the data we used in our experiments. The trademarks mentioned in this paper are the properties of their respective companies.

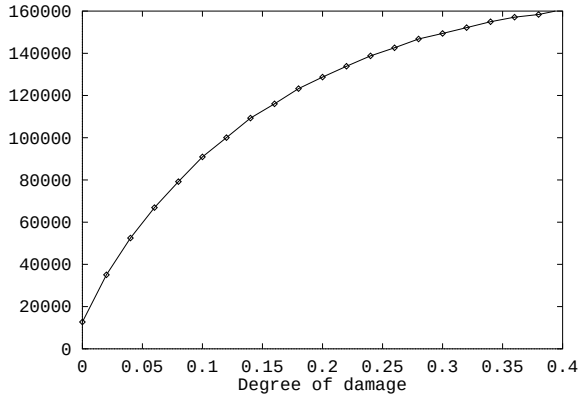

Figure 7: Growth of the index set in the Vector Space model.

References

[1] K. Taghva, J. Borsack, A. Condit, and J. Gilbreth. Results and implications of the noisy data projects. In *Annual Report of UNLV Information Science Research Institute*, pages 49–58, Las Vegas, NV, March 1994.

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

[3] E. Mittehdorf, P. Schäuble, and P. Sheridan. Applying probabilistic term weighting to OCR text in the case of a large alphabetic library catalogue. In *SIGIR'95*, pages 328–335, Seattle, 1995.

[4] 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.

[5] 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.

[6] 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 Third Annual Symposium on Document Analysis and Information Retrieval*, pages 115–126, Las Vegas, NV, April 1994.

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of ACM*, 21:168–173, 1974.

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

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

[17] G. Salton and M. E. Lesk. Computer evaluation of indexing and text processing. *Journal of ACM*, 15(1):8–36, 1968.

[18] J. Savoy. A learning scheme for information retrieval in hypertext. *Information Processing & Management*, 30(4):515–533, 1994.

[19] G. Salton, E. Fox, and U. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(12):1022–1036, 1983.

[20] D. Doermann and S. Yao. Generating synthetic data for text analysis systems. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 449–467, Las Vegas, NV, April 1995.

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

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