

# Improved String Matching Under Noisy Channel Conditions

Kevyn Collins-Thompson<sup>†</sup>  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052 USA  
kevynct@microsoft.com

Charles Schweizer  
Dept. of Electrical and Computer  
Engineering  
Duke University  
Durham, NC 27708 USA  
cbs2@ee.duke.edu

Susan Dumais  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052 USA  
sdumais@microsoft.com

## ABSTRACT

Many document-based applications, including popular Web browsers, email viewers, and word processors, have a “Find on this Page” feature that allows a user to find every occurrence of a given string in the document. If the document text being searched is derived from a noisy process such as optical character recognition (OCR), the effectiveness of typical string matching can be greatly reduced. This paper describes an enhanced string-matching algorithm for degraded text that improves recall, while keeping precision at acceptable levels. The algorithm is more general than most approximate matching algorithms and allows string-to-string edits with arbitrary costs. We develop a method for evaluating our technique and use it to examine the relative effectiveness of each sub-component of the algorithm. Of the components we varied, we find that using confidence information from the recognition process lead to the largest improvements in matching accuracy.

### General Terms

Algorithms, Measurement.

### Keywords

Approximate String Matching, Information Retrieval Evaluation, Noisy Channel Model, Optical Character Recognition.

## 1. INTRODUCTION

In this paper we describe an enhanced version of the standard string search feature available in many document viewing and editing applications. This feature allows the user to find every occurrence of a given word or phrase within a single document. Our algorithm can reliably detect correct matches even when there

are multiple errors in the underlying text, providing a useful increase in recall while maintaining acceptable precision.

This is important for documents whose text is directly obtained from processes such optical character recognition (OCR). Since the recognition process will occasionally mis-recognize letters or combinations of letters with similar shapes, errors appear in the resulting text. Typical error rates on a high-quality image can vary widely depending on the complexity of the layout, scan resolution, and so on. On average, for common types of documents, error rates for OCR are often in the range of 1% to 10% of the total characters on the page.

Our approach, described in detail in section 3, is to pre-filter initial match candidates using an existing fast approximate match procedure. We then score each candidate using an error model based on the noisy channel model of Shannon [SHAN48]. In section 4 we present a technique for evaluating the algorithm at various parameter settings to examine the effectiveness and tradeoffs of our model.

## 2. RELATED WORK

The problem of evaluating and improving retrieval performance on degraded text has been widely studied.

Most of this work has focused on known-item or ranked document retrieval using a pre-computed index. For example, the TREC 4 and 5 OCR confusion tracks [4] and more recent TREC Spoken Document Retrieval evaluations [2], have been the basis for several studies. In general, document retrieval as measured by usual precision and recall methods is fairly robust in the face of OCR recognition errors, assuming relatively good scanned images [13] [15]. This is because a document usually consists of many occurrences of individual words, many of which will be correctly recognized. An extensive analysis of the effect of OCR errors and other types of data corruption on information retrieval can be found in Mittendorf [6]. Specific application examples include the video mail retrieval system of Jones et al. [3], and a spoken document retrieval system developed by Ng [9].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIKM '01, November 5-10, 2001, Atlanta, Georgia, USA.  
Copyright 2001 ACM 1-58113-320-0/00/11 ...\$5.00.

---

<sup>†</sup> Author may now be reached at: School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15232 USA.  
E-mail: kct@cs.cmu.edu

In contrast to document retrieval, we are interested in the situation where we wish to find *every* instance of a word or phrase within a single document very quickly. The “Find on this Page” option in popular Web browsers or Microsoft Office applications are examples of the functionality we wish to support. In our application, recognition errors will cause retrieval failures and thus we need to use approximate matching techniques.

The literature on approximate string matching is extensive. A good overview may be found in [8]. Most approaches use an ‘edit distance’ model of errors in which single character insertions, deletions, and substitutions are allowed, with different costs associated with the different transformations. The popular approximate matching tool *agrep* [14], and the string similarity techniques of Ristad and Yianalos [11] can both accept fairly sophisticated error model descriptions, but are not quite as general as our algorithm, which can use string to string edits with arbitrary costs as well as make use of optional confidence data and language models. Brill et al. use this more general model for automatic spelling correction [1]. Some commercial OCR products such as zyFind™ [16] have incorporated an error-tolerant phrase search based on simple edit distance, but these features are equivalent to just using the first pass of our algorithm. Most of the work on approximate string matching examines the computational complexity of algorithms. There has been comparatively little work that applies the more complex recognition error models and evaluates their accuracy in a systematic fashion as we do in this paper.

### 3. ALGORITHM DESCRIPTION

The algorithm begins with the following inputs:

- A clean query string, without typographical or other errors.
- The document text to be searched, which includes OCR errors.
- An initial threshold value that indicates the error tolerance to be used for finding initial candidates.
- A confusion set describing the most likely types of OCR errors, along with their edit costs.
- Optionally, a table giving the confidence (likelihood of correctness) for each character and/or word in the OCR text.
- A final threshold value used to set the maximum acceptable edit cost.

To find each match, our algorithm passes through four phases, as shown in Figure 1. First, a standard fast approximate-string-match algorithm is used as a pre-filter to obtain match candidates. The key property of this step is that it eliminates unlikely matches very quickly.

Second, once the pre-filter identifies a possible match candidate, we perform a noisy channel analysis, using a dynamic programming algorithm to examine all possible partitioning alternatives to select optimal candidates.

Third, an optional OCR confidence table is used, along with optional word-based heuristics and language model, to adjust the candidate’s score.

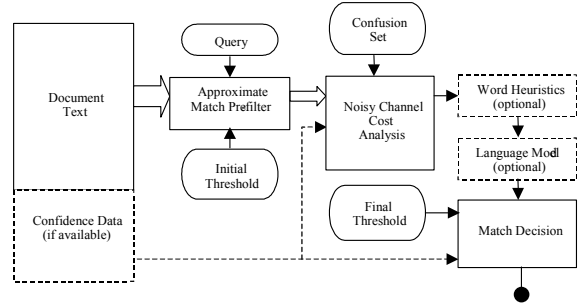


Figure 1. Algorithm Flow Diagram

Finally, we make a final match determination based on whether the candidate’s score exceeded a final threshold.

Each of these phases is described in detail in the following sections.

#### 3.1 Pre-filtering

To find initial match candidates we use a standard approximate string matching routine with a “generous” maximum error threshold. We use a variant of the method developed by Myers in [7]. His algorithm has been optimized for speed and to use minimal storage space.

The final result of this phase is a set of match candidates, each described by a position and a length relative to the input buffer.

#### 3.2 Noisy Channel Cost Analysis

The noisy channel model is one in which source data (for example, the original document text, encoded as an image) is perturbed by noise introduced during a channel process (scanning and OCR), thereby introducing errors into the output data (OCR output text). The result of noisy channel analysis is a probability estimate that a query matches the candidate found in the pre-filter phase.

We model the channel noise introduced during OCR by learning a *confusion set* of typical string-to-string edits. For example, the letter bigram “rn” in the original document is often output as the single letter “m” in the OCR document. Typically the entries in the confusion set comprise combinations of no more than three or four letters, although this is a practical restriction invoked during training, not a limitation of the algorithm.

Each string-to-string edit in the confusion set has an associated probability, namely, the posterior probability  $p(s | R)$ , where  $s$  represents the original string and  $R$  is the corresponding erroneous OCR string. This value is obtained via Bayes Theorem, in which, ignoring the constant denominator,  $p(s | R)$  is given by  $p(R | s) \cdot p(s)$ . The training process by which we obtain estimates for  $p(R | s)$  and  $p(s)$  is described in section 3.3. We take the negative logarithm of  $p(R | s)$  and call this the *edit cost* of the string-to-string edit. It is stored in the confusion set with the corresponding edit entry. A sample confusion set is shown in Table 1.

Given a confusion set  $C$  of  $m$  entries

$$\{s_1 \rightarrow R_1, s_2 \rightarrow R_2, \dots, s_m \rightarrow R_m\}$$

which have corresponding edit costs  $\{c_1, c_2, \dots, c_m\}$ , a query term  $Q$ , and a candidate match  $T$  in the OCR text, we can calculate the probability that  $Q$  matches  $T$  as follows.

First, let us assume that we have already have a partitioning of the query into  $n$  substrings  $\{Q_1, Q_2, \dots, Q_n\}$ , such that for each  $Q_i$ , there is a corresponding set of characters  $T_i$  in  $T$  (possibly empty). If there is more than one possible confusion set entry that matches  $Q_i$  and  $T_i$ , we choose the one with lowest cost. Exactly one of these possibilities is satisfied for each  $Q_i$ :

1.  $Q_i$  maps without errors to its counterpart  $T_i$ , with probability  $p_{CORRECT}(Q_i)$ .
2.  $Q_i$  has an entry in the confusion set such that it maps to  $T_i$  according to the entry  $s_j \rightarrow R_j$ , with probability  $c_i$ .
3.  $Q_i$  maps to some set of characters  $T_i$ , but this mapping is not in the confusion set and hence is modeled by a series of single character insertions, deletions, or substitutions. The probabilities of these operations may vary for individual characters, but for simplicity we denote the overall probabilities as  $p_{INSERT}(Q_i)$ ,  $p_{DELETE}(Q_i)$  and  $p_{SUBST}(Q_i)$  respectively.

If we denote the set of all possible partitions of  $Q$  by  $Part(Q)$ , and assume the transformations are all independent, then we want the most likely of all possible partitions, hence:

$$p(Q | T) = \arg \max_{D \in Part(Q)} \prod_{Q_i \in D} p(Q_i \rightarrow T_i)$$

After expanding the term  $p(Q_i \rightarrow T_i)$  in terms of the probabilities for the possibilities above, and taking the negative logarithm, we have an expression for the total edit cost  $C_{TOTAL}$  for the transformation of  $Q$  to  $T$ :

$$\begin{aligned} C_{TOTAL} = & \arg \min_{D \in Part(Q)} \sum_{Q_a \in D} -\log p_{CORRECT}(Q_a) \\ & + \sum_{Q_b \in D} -\log c_b + \sum_{Q_c \in D} -\log p_{INSERT}(Q_c) \\ & + \sum_{Q_d \in D} -\log p_{DELETE}(Q_d) + \sum_{Q_e \in D} -\log p_{SUBST}(Q_e). \end{aligned}$$

We obtain the most likely partitioning of the query string using a dynamic programming algorithm, setting the costs of  $p_{INSERT}$ ,  $p_{DELETE}$  and  $p_{SUBST}$  using statistics derived from the training phase.

For a concrete example, suppose we are searching for the string “amendment” and come across the document text “arneadme,nt”. For this example we set  $p_{CORRECT}(x) = 0.9$ ,  $p_{INSERT}(x) = p_{SUBST}(x) = 0.1$ ,  $p_{DELETE}(x) = 0.01$  for all strings  $x$ , and use the edit costs from Table 1.

| $s \rightarrow R$      | $-\log p(R   s)$<br>(edit cost) |
|------------------------|---------------------------------|
| am $\rightarrow$ arn   | 1.074                           |
| en $\rightarrow$ ea    | 0.956                           |
| en $\rightarrow$ e,n   | 4.400                           |
| nt $\rightarrow$ at    | 1.013                           |
| end $\rightarrow$ ead  | 0.708                           |
| end $\rightarrow$ eud  | 2.508                           |
| men $\rightarrow$ rnea | 0.858                           |
| me $\rightarrow$ me,   | 1.211                           |

**Table 1. Example Confusion Set**

We have several different ways that the word “amendment” can be partitioned based on this table. For example:

1. am | end | me | nt
2. a | men | d | me | nt

In the first case above, the total edit cost to transform “am | end | me | nt” into the corresponding OCR strings “arn | ead | me, | nt” would be calculated as follows.

$$\begin{aligned} C_1 = & -\log p(\text{am} \rightarrow \text{arn}) - \log p(\text{end} \rightarrow \text{ead}) - \log p(\text{me} \rightarrow \text{me,}) - \log \\ & p_{CORRECT}(\text{nt} \rightarrow \text{nt}) \\ = & 1.074 + 0.708 + 1.211 + 0.105 = \mathbf{3.098} \end{aligned}$$

Compare this to the optimal partitioning, “a | men | d | me | nt”, which gives:

$$\begin{aligned} C_2 = & -\log p_{CORRECT}(a \rightarrow a) - \log p(\text{men} \rightarrow \text{rnea}) - \log p_{CORRECT} \\ & (d \rightarrow d) - \log p(\text{me} \rightarrow \text{me,}) - \log p_{CORRECT}(\text{nt} \rightarrow \text{nt}) \\ = & 0.105 + 0.858 + 0.105 + 1.211 + 0.105 = \mathbf{2.384} \end{aligned}$$

These edit cost values are passed to the third phase for possible adjustment before the comparison to the final threshold.

With the final threshold set to 0.300, and 9 characters in the query term, the threshold for this query is  $9 \cdot 0.300 = 2.700$ . If no other modifications to the final threshold or costs are made, case 2 would be considered a valid match since the candidate cost of 2.384 is less than the final threshold of 2.700. Case 1 would not be considered a valid match since its score of 3.098 is greater than the final threshold of 2.700.

### 3.3 Training the Model

To train our noisy channel error model we selected a subset of files not used in the evaluation, amounting to approximately 20% of the total text size in each corpus.

Using this sub-collection, we ran a processing pass similar to that used for evaluation in section 4, but using a high error tolerance – typically with an expected error rate of 3 errors every 4 characters. Using the syntactic signature method described in Section 4, we extract the correct matches from these results to get a set of pairs  $(S, T)$  where  $S$  is a word from the ground-truth file and  $T$  is the corresponding noisy OCR word.

For each (S, T) pair, we found the greatest common substrings between S and T, from which we derived an initial set of possible edits. We then expanded this set using up to 3 characters of context on either side of the edit, for both the ground-truth word and the OCR word. For each edit  $s \rightarrow R$  in this expanded set, we kept track of the edit's overall frequency, the frequency of all other edits based on  $s$ , and the total frequency of  $s$  in the corpus. From this we calculate  $p(R | s)$  and thus the edit cost

$$c = -\log p(R | s).$$

We also calculate  $p(s)$  and then select the most useful edits – those with the highest values of  $p(R | s) \cdot p(s)$ . For our experiments we kept the top 2500 edits.

### 3.4 Optional Processing

In the third phase, we make use of confidence information and heuristics to adjust the candidate's edit cost.

#### 3.4.1 Word Heuristics

Since users often search on one or more complete words, the algorithm can be modified to include position-based probabilities that reflect the importance that a match be close to a complete word or word prefix. For our experiments we tested for either punctuation or whitespace at the start and end of a match, and reduced error costs by 0.25 for a word prefix match and 0.50 for an entire word match.

#### 3.4.2 Language Models

Even when we do not have confidence information from the recognizer, we can calculate a rough confidence estimate based on simple language models. Since our documents were in English, we used a frequency table of English bigrams (obtained on a separate training corpus) and gave either a “low” or “high” confidence estimate to any words containing at least one “rare” bigram or none, respectively.

#### 3.4.3 Using Confidence Data

If the recognition process provides character or word-level confidence data, we can use this information. We do this by increasing edit costs in the noisy channel model according to a region's confidence value. For high-confidence regions, this essentially reduces to performing exact matching. The recognition engine may sometimes give an indeterminate confidence value for a word, in which case the language model may be optionally invoked to supply an estimate. Ideally, we would make use of character-level confidence data, and plan to do so in future versions. Our current implementation only stores word-level confidence data in the document to reduce the file size.

### 3.5 Match Decision

In the fourth phase, we compare the final threshold against the match candidate's score. If the candidate's score is above the final threshold, it is not counted as a match.

## 4. EVALUATION

### 4.1 Methodology

We compare baseline word matching performance with our algorithm using various sub-components such as the OCR confusion set, word heuristics, and word-level confidence.

We ran experiments using two different test collections.

1. A subset of 5 documents from the TREC-5 confusion track corpus [4]. These documents are from the 1994 Federal Register and contain about 160,000 words. This text has no confidence data available, and the OCR conversion was done by NIST.

2. A collection of 200 document images with ground truth text and corresponding OCR text, containing approximately 100,000 English word occurrences. The OCR text contains word-level confidence scores and was generated using an OCR engine licensed from Scansoft, Inc.

We perform whitespace-delimited word-breaking and extract stopwords from the ground truth files to obtain a list of query terms. For each document, we ask every unique word in the ground truth document as a ‘query’, giving us approximately 8,600 queries from the first test collection and 15,000 from the second. We assume that the query is clean, containing no spelling or typographical errors. There are scenarios where our algorithm could be useful with queries containing errors but we do not address those in this paper.

We want to measure how accurately we find each query in the OCR document. We define a ‘true match’ as a string in the OCR'd document that matches the corresponding query term in the ground truth document. Occasionally, mismatches in the original document are corrupted by the OCR process into strings that match in the OCR'd document; these are ‘false matches’. Any query, for which a word exists in the ground truth document, but which fails to find the corresponding word in the OCR document, is termed a ‘miss’.

If the numbers of true matches, false matches, and misses are  $t$ ,  $f$ , and  $m$  respectively, then precision  $p$  and recall  $r$  are derived using the formulas:

$$p = \frac{t}{t + f}, \quad r = \frac{t}{t + m}$$

We also report van Reijsbergen's F-measure [10] to provide a single number that combines precision  $p$  and recall  $r$  for evaluation purposes. This is given by:

$$F(\alpha) = \frac{pr}{\alpha r + (1 - \alpha) p}$$

The relative importance given to precision versus recall is expressed through the parameter  $\alpha$ . When  $\alpha = 0.5$ , they are given equal weight. We include two different values of  $\alpha$ , corresponding to neutral ( $\alpha = 0.5$ ) preferences, and recall-oriented ( $\alpha = 0.2$ ), which we believe to be important in ‘find on this page’ applications.

The key problem in evaluating string-matching results on degraded text is obtaining a reliable correspondence between words in the ground truth file and their noisy counterparts. A

single word in the original document has a counterpart in the OCR document, but it may not be obvious what it is. We need to be able to identify these correspondences in order to distinguish ‘true matches’ and ‘false matches’. Ideally, we would compare the geometric positions of the corresponding words in the image, but this kind of positional data is not currently available in either of our test sets.

To solve this problem, we construct a syntactic signature for each word using  $N$  non-whitespace characters immediately leading or following the word. In practice we use a value for  $N$  of 20. To test if a word occurrence in the truth file is the same as one occurring in the degraded file, their signatures are compared according to a simple edit distance. In this matching process we allow a relative error tolerance that is twice the average OCR error rate. While this signature is theoretically not unique, in practice it works very well to compare word occurrences quickly. We search for each query term in both the ground truth file and the corresponding OCR file and compare the two result lists using the syntactic signature. With this matching, we can identify corresponding words in the ground truth and OCR documents.

## 4.2 Results for TREC-5 Confusion Track Files

The measurements on our subset of the TREC confusion corpus are shown below in Table 2. The exact matching score is the accuracy obtained by exact matching of the query string to the OCR text. The baseline measurement uses only the first phase of our algorithm to perform simple approximate matching without an error model. We were primarily interested in the effectiveness of adding a trained error model for edit costs and a general language model for estimating word-level confidence, and the relative contributions of each. We evaluated the effectiveness of adding the word heuristic but found no differences, so the results are omitted here for simplicity. We also varied the final match threshold to allow for different error cost tolerances. The best-performing parameters for each F value are shown in bold.

Because the number of queries is large, even small differences in F are statistically significant. For the TREC collection, the average standard error about the mean is .000295, so differences that are .0005782 or larger are significant at the .05 level; for the In-House collection, the average standard error about the mean is .0000856, so differences that are .0001678 or larger are significant at the .05 level. We focus our discussion on differences that are the most interesting theoretically or practically.

Using a trained error model results in a higher value of both types of F value for threshold values of 0.200 or less. For recall-oriented F, the error model gives a useful improvement at threshold levels of 0.300 and below. Adding the bigram language model to the trained error model improves performance for threshold values of 0.300 and 0.400. For recall-oriented F, the best performance (0.936) is achieved using a fairly high threshold with the two error correction techniques combined. The F value of 0.936 represents a 15% improvement over exact matching (0.812) and a 25% improvement over a simple approximate match approach.

In general, an effective strategy is to pair sub-components that mainly boost recall and expand the set of initial candidates with

those that improve precision and are good at eliminating false matches. For example, addition of the language model gave fairly consistent improvements in overall accuracy when paired with a trained error model or a higher threshold.

We also examined the matching accuracy at various query lengths. An example of these results for a typical document is given in Figure 2. The high final threshold of 0.900 was used to highlight the change in precision. Precision of the matches generally improved as the length of the query increased. This result is consistent with the fact that for longer English words there are fewer words that are “close” in terms of edit distance, and thus there are fewer potential mismatches. The results suggest that using exact matching or a higher error threshold may be appropriate for shorter queries, when searching documents with word error rates comparable to the values we studied.

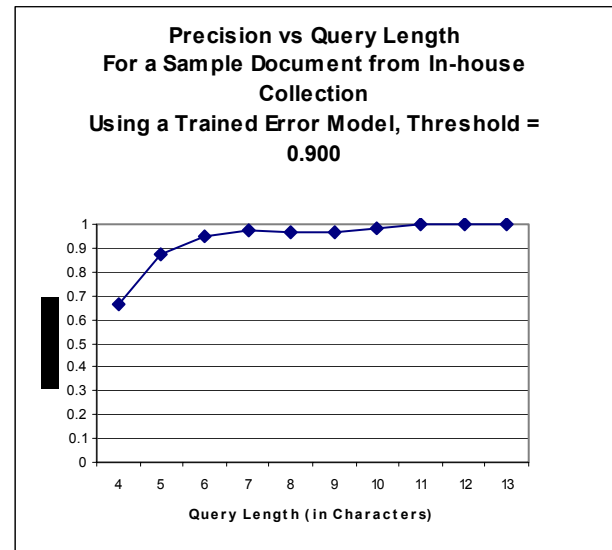


Figure 2. Change in Precision with Query Length

## 4.3 Results for In-House Collection, with Confidence Information

For our in-house collection of documents, we had word-level confidence data available from the OCR engine. This is typically a value between 0 (lowest confidence) and 1000 (highest confidence). For the purposes of string matching, the confidence value for any sub-string of a word is approximated by the confidence value of the word itself.

For this collection we examined the effect of a learned error model, a language model for estimating confidence, and word-level confidence available from the recognizer. We also looked at different final match thresholds and report precision, recall, neutral-F and recall-oriented F measures. The results are presented in Table 3.

Incorporating confidence information gave consistently higher overall accuracy – more than any other single component. Using confidence information alone provides the largest gains, and adding the learned error model provides some additional small advantages for the neutral F and for recall-oriented F at the lowest threshold.

Table 2. TREC-5 Subset: Word Error Rate = 20%

|                               | Precision | Recall | Neutral-<br>F | % Gain<br>Over<br>Baseline | Recall-<br>Oriented<br>F | % Gain<br>Over<br>Baseline |
|-------------------------------|-----------|--------|---------------|----------------------------|--------------------------|----------------------------|
| Exact matching                | 0.998     | 0.775  | 0.873         |                            | 0.812                    |                            |
| Baseline: threshold = 0.100   | 0.980     | 0.800  | 0.881         |                            | 0.831                    |                            |
| +error model                  | 0.992     | 0.867  | 0.925         | 4.99%                      | 0.889                    | 6.98%                      |
| +error model, +language model | 0.994     | 0.864  | 0.924         | 4.88%                      | 0.887                    | 6.74%                      |
| Baseline: threshold = 0.200   | 0.947     | 0.857  | 0.900         |                            | 0.874                    |                            |
| +error model                  | 0.890     | 0.910  | 0.900         | 0.00%                      | 0.906                    | 3.66%                      |
| +error model, +language model | 0.827     | 0.913  | 0.868         | -3.56%                     | 0.894                    | 2.29%                      |
| Baseline: threshold = 0.300   | 0.488     | 0.867  | 0.624         |                            | 0.750                    |                            |
| +error model                  | 0.750     | 0.940  | 0.834         | 33.65%                     | 0.895                    | 19.33%                     |
| +error model, +language model | 0.850     | 0.960  | 0.902         | 44.55%                     | 0.936                    | 24.80%                     |
| Baseline: threshold = 0.400   | 0.56      | 0.950  | 0.705         |                            | 0.834                    |                            |
| +error model                  | 0.510     | 0.980  | 0.671         | -4.82%                     | 0.827                    | -0.84%                     |
| +error model, +language model | 0.696     | 0.980  | 0.814         | 15.46%                     | 0.906                    | 8.63%                      |

TABLE\_3\_GOES\_HERE:      FORMATTING      PROBLEMS!

## 5. CONCLUSIONS

We described an enhanced string-matching algorithm for degraded text. The algorithm is more general than standard approximate matching algorithms, allowing string-to-string edits with arbitrary costs as well as the use of confidence information. We develop a method for evaluating our technique and use it to examine the relative effectiveness of each sub-component of the algorithm. Based on our experiments, we can draw the following conclusions.

First, for users who weight recall more highly than precision, our algorithm provides improvement over both exact match and standard approximate matching without a trained error model. For users with neutral precision/recall preferences, the benefits are smaller but still measurable. The improvements were largest for the TREC collection, which had a higher word error rate.

Second, using confidence data from the OCR engine, when available, results in the largest improvement in matching accuracy, compared to using a simple language model or trained error model alone.

Third, we recommend that the algorithm be applied to longer query terms only, such as those more than four characters long, with exact match or a higher error threshold being used for shorter query terms.

In general, although our algorithm depends on having trained confusion sets and confidence information to achieve its best performance, we believe this is entirely appropriate for applications that rely on a specific recognition engine.

In future work, we intend to use character-level confidence information in the noisy channel analysis, and we will continue to explore methods for training better error models. We also intend to apply this approach to the output from handwriting recognition, and to include languages other than English in our analysis.

## 6. ACKNOWLEDGEMENTS

The authors would like to thank Henry Burgess, Rado Nickolov, Stephen Robertson, and an anonymous reviewer for their helpful suggestions.

## 7. REFERENCES

- [1] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," Proceedings of ACL 2000, Hong Kong, Oct. 2000.
- [2] J. Garofolo, C. Auzanne, and E. Voorhees, "The TREC spoken document retrieval track: A success story," in *Text Retrieval Conference (TREC) 8*, E. Voorhees, Ed., Gaithersburg, Maryland, USA, 1999, November 16- 19.
- [3] G. Jones, J. Foote, K. S. Jones, and S. Young, "Video mail retrieval: The effect of word spotting accuracy on precision," in Proc. ICA SSF '95, Detroit, MI, pp. 309—312, May 1995.
- [4] P. Kantor, and E. M. Voorhees, "The TREC-5 Confusion Track: Comparing Retrieval Methods for Scanned Text", Information Retrieval, vol. 2, pp. 165—176, 2000.
- [5] K. Marukawa, T. Hu, H. Fujisawa, and Y. Shima, "Document retrieval tolerating character recognition errors — evaluation and application," Pattern Recognition, vol. 30, no. 8, pp. 1361—1371, 1997.
- [6] E. Mittendorf. *Data Corruption and Information Retrieval*. PhD thesis, ETH Zürich, Institute of Computer Systems, January 1998.  
<ftp://ftp.inf.ethz.ch/pub/publications/dissertations/th12507.ps.gz>
- [7] G. Myers. A fast bit-vector algorithm for approximate pattern matching based on dynamic programming. In Proc Combinatorial Pattern Matching 98, 1-13, Springer-Verlag, 1998.
- [8] G. Navarro. Approximate Text Searching. PhD thesis, Dept. of Computer Science, Univ. of Chile, December 1998. Technical Report TR/DCC-98-14. <ftp://ftp.dcc.uchile.cl/pub/-users/gnavarro/thesis98.ps.gz>.
- [9] K. Ng and V. Zue, "Subword unit representations for spoken document retrieval," in Proc. Eurospeech '97, Rhodes, Greece, pp. 1607—1610, Sept. 1997.
- [10] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2<sup>nd</sup> edition, 1979.
- [11] E. S. Ristad, P. N. Yianilos, "Learning String Edit Distance," Princeton University Research Report CS-TR-532-96, 1996.
- [12] C. Shannon, "A mathematical theory of communication," Bell System Technical Journal 27(3): 379 – 423, 1948.
- [13] K. Taghva, J. Borsack, A. Condit, and S. Erva. The effects of noisy data on text retrieval. In *UNLV Information Science Research Institute Annual Report*, pages 71 – 80, 1993.
- [14] S. Wu and U. Manber, "Fast Text Searching Allowing Errors," Communications of the ACM, vol. 35, no. 10, pp 83—91, 1992.
- [15] C. Zhai, X. Tong, N. Milic-Frayling, and D. Evans, "OCR correction and expansion for retrieval on OCR data — CLARIT TREC-5 confusion rack report," in Proceedings of Fifth Text REtrieval Conference (TREC-5), Gaithersburg, MD, USA, Nov. 1996. NIST-SP 500-238.
- [16] ZyFind™ – A subsystem of ZylImage™, ZyLAB International Inc., Rockville, MD. <http://www.zylab.com/>