

# Performance Evaluation for Text Processing of Noisy Inputs

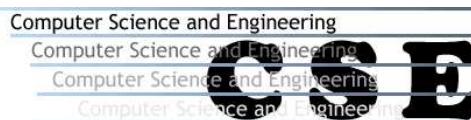
*Daniel Lopresti*

March 2005

Technical Report LU-CSE-05-011

Department of Computer Science and Engineering  
Lehigh University  
Bethlehem, PA 18015 USA

<http://www.cse.lehigh.edu/>



# Performance Evaluation for Text Processing of Noisy Inputs\*

*Daniel Lopresti*

lopresti@cse.lehigh.edu

Department of Computer Science and Engineering

Lehigh University

Bethlehem, PA 18015

March 2005

## Abstract

We investigate the problem of evaluating the performance of text processing algorithms on inputs that contain errors as a result of optical character recognition. A new hierarchical paradigm is proposed based on approximate string matching, allowing each stage in the processing pipeline to be tested, the error effects analyzed, and possible solutions suggested.

Keywords: *performance evaluation, optical character recognition, sentence boundary detection, tokenization, part-of-speech tagging.*

## 1 Introduction

Increasingly, researchers are acknowledging that many real-world sources are “noisy” and require attention to developing techniques that are robust in the presence of such noise. The outputs from optical character recognition (OCR) and automatic speech recognition (ASR) systems, for example, typically contain various degrees of errors, and even purely electronic (“born digital”) media, such as email, are not error-free. To exploit these documents, we need to develop techniques to deal with noise, in addition to working on core text processing issues. Whether we can successfully handle noise will greatly influence the ultimate utility of the information extracted from such documents.

A number of researchers have begun studying problems relating to information extraction from noisy sources. To date, this work has focused predominately on errors that arise during speech recognition. For example, Gotoh and Renals propose a finite state modeling approach to extract sentence boundary information from text and audio sources, using both  $n$ -gram and pause duration information [GR00]. They found that precision and recall of over 70% could be achieved by combining both kinds of features. Palmer and Ostendorf describe an approach for improving named entity extraction by explicitly modeling speech

---

\*Presented at *20th Annual ACM Symposium on Applied Computing (SAC2005), Document Engineering Track*, Sante Fe, NM, March 2005.

recognition errors through the use of statistics annotated with confidence scores [PO01]. Hori and Furui summarize broadcast news speech by extracting words from automatic transcripts using a word significance measure, a confidence score, linguistic likelihood, and a word concatenation probability [HF01].

There has been much less work, however, in the case of noise induced by optical character recognition. Early papers by Taghva, Borsack, and Condit show that moderate error rates have little impact on the effectiveness of traditional information retrieval measures [TBC96a, TBC96b], but this conclusion seems specific to certain assumptions about the IR model (“bag of words”), the OCR error rate (not too low), and the length of the documents (not too short). Miller, *et al.* study the performance of named entity extraction under a variety of scenarios involving both ASR and OCR output [MBS<sup>+</sup>00], although speech is their primary interest. They found that by training their system on both clean and noisy input material, performance degraded linearly as a function of word error rates. They also note in their paper: “To our knowledge, no other information extraction technology has been applied to OCR material” (pg. 322).

A recent paper by Jing, Lopresti, and Shih studied the problem of summarizing textual documents that had undergone optical character recognition and hence suffered from typical OCR errors [JLS03]. In that study, the focus was on analyzing how the quality of summaries was affected by the level of noise in the input document, and how each stage in summarization was impacted by the noise. Based on this analysis, the authors suggested possible ways of improving the performance of automatic summarization systems for noisy documents. From the standpoint of performance evaluation, however, this work did not provide rigorous criteria, instead employing a variety of indirect measures: for example, comparing the total number of sentences returned by sentence boundary detection for clean and noisy versions of the same input text, or counting the number of incomplete parse trees generated by a part-of-speech tagger.

Performance evaluation is a challenging issue that is key to developing robust text processing algorithms. In this paper, we focus directly on the former as a subject worthy of study in a context independent of (and hence not subservient to) the latter.

We begin in Section 2 by describing the typical stages in a text processing system. Rather than assume a specific final application (*e.g.*, summarization or named-entity extraction), we restrict our focus to procedures that are common to a variety of problem areas. We then propose in Section 3 an evaluation paradigm based on the hierarchical application of approximate string matching techniques using dynamic programming. This flexible yet mathematically rigorous approach allows us both to quantify the performance of a given text processing stage as well as to identify explicitly the errors it has made. Section 4 presents the results of a pilot study we performed in which we selected a small set of documents and created noisy versions of them. These were then OCR’ed and piped through procedures for performing sentence boundary detection, tokenization, and part-of-speech tagging. The experimental results show that these modules suffer significant degradation as the noise level in the document increases, but, more importantly, that the performance evaluation paradigm we are developing can provide useful assistance in such analyses. We conclude with a proposal for future work.

## 2 Stages in Text Processing

In this section, we describe in general terms the stages that are common to many text processing systems, and then list the specific packages we have tested in a preliminary study of our paradigm for performance evaluation. The stages, in order, are typically:

1. Optical character recognition.
2. Sentence boundary detection.
3. Tokenization.
4. Part-of-speech tagging.

In selecting implementations of the above procedures, we elected to employ freely available open source software rather than proprietary, commercial solutions.<sup>1</sup> While the programs we tested are not necessarily state-of-the-art, they offer other significant benefits. From the standpoint of evaluation, all we require is behavior that is representative, not “best-in-class.” Our methodology should apply equally well to other approaches to solving the same problems, no matter what algorithms are used.

### 2.1 Optical Character Recognition

The first stage of the pipeline is optical character recognition, the conversion of the scanned input image from bitmap format to encoded text. Optical character recognition performs quite well on clean inputs in a known font. It rapidly degrades in the case of degraded documents, complex layouts, and/or unknown fonts. The input to OCR is an image in bitmap format, and its output is the recognized text, possibly including errors, in an encoded format (*e.g.*, ASCII). In certain situations, OCR will introduce many errors involving punctuation characters, which has an impact on later-stage processing.

For our OCR stage, we selected the open source *gocr* package [S<sup>+</sup>04]. While *gocr* is not competitive with commercial systems in terms of performance, it is freely available and easily integrated into a batch mode text processing pipeline. Since we are presenting it with relatively simple text layouts, having to deal with complex inputs is not an issue. The performance of *gocr* on the inputs we tested is likely to be similar to the performance of a better-quality OCR package on noisier inputs of the same type. Hence, the conclusions we draw here almost certainly apply to other OCR systems, provided the input is sufficiently noisy.

### 2.2 Sentence Boundary Detection

Procedures for sentence boundary detection use a variety of syntactic and perhaps semantic cues in order to break the input text into sentence-sized units, one per line (*i.e.*, each unit is terminated by a standard end-of-line delimiter such as the Unix newline character).

---

<sup>1</sup>In the spirit of this same philosophy, we plan to make our code for performance evaluation available via open source as well once it has been thoroughly debugged and vetted.

The sentence boundary detector we used in our test is the MXTERMINATOR package by Reynar and Ratnaparkhi [RR97].

### 2.3 Tokenization

Tokenization takes the input text, which has been divided into one sentence per line, and breaks it into individual tokens which are delimited by white space. These largely correspond to word-like units or isolated punctuation symbols. In our studies, we used the Penn Treebank tokenizer [Mac95]. As noted in the documentation for that system, its operation can be summarized as:

- most punctuation is split from adjoining words,
- double quotes are changed to doubled single forward- and backward-quotes,
- verb contractions and the Anglo-Saxon genitive of nouns are split into their component morphemes, and each morpheme is tagged separately.

### 2.4 Part-of-Speech Tagging

Part-of-speech tagging takes the tokenized text as input and tags each token as per its part of speech. We used Ratnaparkhi’s part-of-speech tagger MXPOST [Rat96], which produced a total of 42 different part-of-speech tags for our test inputs.

## 3 Performance Evaluation

As noted previously, performance evaluation for text processing of noisy inputs presents some serious challenges. Our goal in pursuing this work is to develop such techniques and demonstrate their utility. For the approach we are proposing, one body of past work is particularly relevant: that is, the use of approximate string matching to align two linear streams of text, one representing OCR results and the other representing the ground-truth [ELS94, ELSZ94].

This optimization problem can be solved using a well-known dynamic programming algorithm [NW70, WF74]. Let  $S = s_1s_2 \dots s_m$  be the source document (the ground-truth),  $T = t_1t_2 \dots t_n$  be the target document (the OCR results), and define  $dist1_{i,j}$  to be the distance between the first  $i$  symbols of  $S$  and the first  $j$  symbols of  $T$ . The initial conditions are:

$$\begin{aligned} dist1_{0,0} &= 0 \\ dist1_{i,0} &= dist1_{i-1,0} + c1_{del}(s_i) \\ dist1_{0,j} &= dist1_{0,j-1} + c1_{ins}(t_j) \end{aligned} \tag{1}$$

and the main dynamic programming recurrence is:

$$dist1_{i,j} = \min \begin{cases} dist1_{i-1,j} + c1_{del}(s_i) \\ dist1_{i,j-1} + c1_{ins}(t_j) \\ dist1_{i-1,j-1} + c1_{sub}(s_i, t_j) \end{cases} \tag{2}$$

for  $1 \leq i \leq m$ ,  $1 \leq j \leq n$ . Here deletions, insertions, and mismatches are charged positive costs, and exact matches are charged negative costs. The computation builds a matrix of distance values working from the upper left corner ( $dist1_{0,0}$ ) to the lower right ( $dist1_{m,n}$ ).

By maintaining the decision(s) used to obtain the minimum in each step, it becomes possible to backtrack the computation and obtain, in essence, an explanation of the errors that arose in processing the input. Such information can be invaluable in analyzing the performance of text processing algorithms.

This basic approach to computing edit distance can be supplemented to model split and merge operations, which we can think of as multi-symbol substitutions. For example,  $c_{sub_{k:l}}$  corresponds to a substitution of  $k$  symbols in the source string to  $l$  symbols in the target string. The recurrence then becomes:

$$dist1_{i,j} = \min \begin{cases} dist1_{i-1,j} + c1_{del}(s_i) \\ dist1_{i,j-1} + c1_{ins}(t_j) \\ \min_{1 \leq k' \leq k, 1 \leq l' \leq l} [ dist1_{i-k',j-l'} + \\ c1_{sub_{k:l}}(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) ] \end{cases} \quad (3)$$

where  $k'$  and  $l'$  range up to the maximum allowed size of a multi-symbol substitution.

In looking at how these ideas might be generalized to later stages of text processing, we consider the output of those stages and the errors that might arise. Tokenization, for example, might fail to recognize a token boundary thereby combining two tokens into one (a “merge”), or break a token into two more more pieces (a “split”). Similar errors may arise in sentence boundary detection.

While a single level of dynamic programming is sufficient to account for such effects, it will not allow us to reconstruct what has happened at a high level and assign errors to the processing stages where they arose. To accomplish this goal, we need to apply additional levels in the optimization [Lop01].

In our current work, we adopt a three level hierarchy. At the highest level, sentences (or purported sentences) are matched allowing for missed or spurious sentence boundaries. The basic entity in this case is a sentence string, and the costs of deleting, inserting, substituting, splitting, or merging sentence strings are defined recursively in terms of the next level of the hierarchy, which is tokens. As with the sentence level, tokens can be split or merged. Comparison of tokens is defined in terms of the lowest level of the hierarchy, which is the basic approximate string matching model we began this section with (Equations 1-3).

In terms of dynamic programming, at the token level, the algorithm becomes:

$$dist2_{i,j} = \min \begin{cases} dist2_{i-1,j} + c2_{del}(s_i) \\ dist2_{i,j-1} + c2_{ins}(t_j) \\ \min_{1 \leq k' \leq k, 1 \leq l' \leq l} [ dist2_{i-k',j-l'} + \\ c2_{sub_{k:l}}(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) ] \end{cases} \quad (4)$$

where the inputs are assumed to be sentences and  $c_{del}$ ,  $c_{ins}$ , and  $c_{sub}$  are now the costs of deleting, inserting, and substituting whole tokens, respectively. The initial conditions are defined analogously to Equation 1.

Since the basic editing operations now involve tokens, it is natural to define the new costs as:

$$\begin{aligned} c2_{del}(s_i) &\equiv dist1(s_i, \phi) \\ c2_{ins}(t_j) &\equiv dist1(\phi, t_j) \\ c2_{sub_{k:l}}(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) &\equiv dist1(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) \end{aligned} \quad (5)$$

where  $\phi$  is the null string. Hence, the second-level computation is defined in terms of the first-level computation.

Lastly, at the highest level, the input is a whole page and the basic editing entities are sentences. For the recurrence, we have:

$$dist3_{i,j} = \min \begin{cases} dist3_{i-1,j} + c3_{del}(s_i) \\ dist3_{i,j-1} + c3_{ins}(t_j) \\ \min_{1 \leq k' \leq k, 1 \leq l' \leq l} [dist3_{i-k',j-l'} + \\ c3_{sub_{k:l}}(s_{i-k'+1\dots i}, t_{j-l'+1\dots j})] \end{cases} \quad (6)$$

with costs:

$$\begin{aligned} c3_{del}(s_i) &\equiv dist2(s_i, \phi) \\ c3_{ins}(t_j) &\equiv dist2(\phi, t_j) \\ c3_{sub_{k:l}}(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) &\equiv dist2(s_{i-k'+1\dots i}, t_{j-l'+1\dots j}) \end{aligned} \quad (7)$$

By executing this hierarchical dynamic programming from the top down, given an input page for the OCR results and another for the ground-truth, we can determine an optimal alignment between purported sentences, which is defined in terms of an optimal alignment between individual tokens in the sentences, which is defined in terms of an optimal alignment between each possible pairing of tokens (including the possibilities that tokens are deleted, inserted, split, or merged). Once an alignment is constructed using the orthography of the input text strings, we may compare the part-of-speech tags assigned to corresponding tokens to study the impact of OCR errors on that process as well.

## 4 Experimental Results

We took 10 pages of text from the opening chapters of the Project Gutenberg edition of the well-known novel *Moby-Dick* [Pro04], formatted the pages in 12-point Times, printed them on a laserprinter, and then scanned them at 300 dpi bitonal using an automatic sheet feeder. One set of pages was scanned as-is, another was first photocopied through three generations with the contrast set to the darkest possible setting, a third was similarly photocopied through three generations at the lightest possible setting, and a fourth set was faxed before scanning. We then ran the resulting bitmap images through the *gocr* OCR package. Examples of small regions of scanned page images and OCR output appear in Figures 3-6.

It is traditional to judge basic OCR accuracy using a single level of dynamic programming, *i.e.*, Equations 1-2 (see, *e.g.*, [ELS94]). These results for the four datasets are presented in Table 1. As in the information retrieval domain, precision and recall reflect two different aspects of system performance. The former is the fraction of reported entities that are true, while the latter is the fraction of true entities that are reported. Note that the

	All Symbols			Punctuation			Whitespace		
	Prec.	Recall	Overall	Prec.	Recall	Overall	Prec.	Recall	Overall
Clean	0.982	0.988	0.988	0.843	0.973	0.912	0.974	0.996	0.985
Light	0.646	0.747	0.790	0.193	0.648	0.315	0.589	0.965	0.730
Dark	0.411	0.575	0.628	0.090	0.686	0.170	0.391	0.884	0.539
Fax	0.584	0.644	0.732	0.201	0.625	0.306	0.608	0.890	0.717

Table 1: Average OCR performance relative to ground-truth.

baseline OCR accuracy is quite high, but performance deteriorates rapidly for the degraded documents. It is also instructive to consider separately the impact on punctuation symbols and whitespace; these results are also shown in the table. Note that punctuation symbols in particular are badly impacted, with a large number of false alarms (low precision), especially in the case of the dark dataset where fewer than one in 10 reports are true. This phenomenon has serious implications for sentence boundary detection and later stages of text processing.

We then ran sentence boundary detection, tokenization, and part-of-speech tagging on both the OCR output and the ground-truth and compared the results using the paradigm described in the previous section. As noted, this allows us to both quantify performance as well as to determine the optimal alignments between sequences and hence identify the actual errors that have arisen. An example of a relatively simple alignment taken from the light dataset is shown in Figure 1. On the other hand, Figure 2 shows a more challenging

VB	IN	DT	NNS	IN	NNS	RB	.
Look	at	the	crowds	of	water-gazers	there	.
Ground-Truth							
NN	IN	DT	, NNS	IN	NNS	RB	.
_oo_	at	the	, rowds	of	water-gazers	th_re	.
				OCR Output			

Figure 1: Example of a straightforward alignment.

instance from the same dataset, demonstrating the power of this technique in aligning very noisy inputs.

A tabulation of the dynamic programming results for the three text processing stages appears in Table 2. Here we see that the clean input is processed with better than 95% accuracy, which is what we would expect. Note, however, that the degraded documents yield poor results, especially the dark dataset which was flagged earlier for the large number of spurious punctuation symbols it introduces. As our work in this area is just beginning, we expect to be able to develop more detailed analyses of such results, on larger datasets, as time progresses.

IN	PRP	CC	VBD	PRP	,	RB	DT	NNS	IN	PRPS		
If	they	but	knew	it	,	almost	all	men	in	their		
Ground-Truth												
IN	NN	CC	NN	: NNP	,	VBP	DT	NNP	,	:	NNP	: DT
If	theY	but	_new	; t	,	a_Most	all	Me	,	;	,	the ; r
OCR Output												
NN	,	DT	NN	CC	JJ	,	JJ	RB	RB			
degree	,	some	time	or	other	,	cherish	very	nearly			
Ground-Truth												
NN	,	DT	NN	CC	JJ	,	JJ	NN	, NNP	,	VBP	
degree	,	some	time	or	other	,	chejsh	ve.	, y	,	e__y	
OCR Output												
DT	JJ	NNS	IN	DT	NN	IN	PRP	.				
the	same	feelings	towards	the	ocean	with	me	.				
Ground-Truth												
DT	JJ	NN	, VBZ	RB	DT	NN	IN	PRP	.			
the	same	feeli	, gs	towards	the	ocean	with	me	.			
OCR Output												

Figure 2: Example of a challenging alignment.

## 5 Conclusions

In this paper, we have discussed some of the challenges in evaluating the performance of text processing algorithms operating on noisy documents. In particular, we considered a pipeline consisting of four stages: optical character recognition, sentence boundary detection, tokenization, and part-of-speech tagging. Our ultimate goal is to study methodologies for testing each step when run on noisy documents and analyzing the errors that arise. Such knowledge will provide a solid basis for the development of more robust text processing techniques.

Many followup investigations suggest themselves, some of which have already been suggested in the context of earlier work on summarization [JLS03]. We regard the user interface as a crucial component in real-world text processing systems. Given that noisy documents, and hence data extracted from them, may contain errors, it is important to find the best ways of displaying such information so that the user may proceed with confidence, knowing that the information is truly representative of the document(s) in question.

Since errors propagate from one stage of the pipeline to the next, sentence boundary detection algorithms that work reliably for noisy documents are clearly important. One way to achieve this might be to retrain an existing system on noisy documents so that it will be more tolerant of noise. However, this is only applicable if the noise level is low.

	Sentence Boundaries			Tokenization			Part-of-Speech Tagging		
	Prec.	Recall	Overall	Prec.	Recall	Overall	Prec.	Recall	Overall
Clean	0.939	0.985	0.961	0.975	0.994	0.984	0.953	0.975	0.964
Light	0.648	0.906	0.731	0.646	0.877	0.733	0.307	0.500	0.380
Dark	0.321	0.995	0.405	0.388	0.691	0.479	0.097	0.210	0.132
Fax	0.442	0.987	0.536	0.494	0.674	0.563	0.203	0.303	0.242

Table 2: Average text processing performance relative to ground-truth.

Significant work is needed to develop robust methods that can handle documents with high noise levels.

It is important to choose an appropriate unit level to represent information. For clean text, sentence extraction is a feasible goal since we can reliably identify sentence boundaries. For documents with very low levels of noise, sentence extraction is still possible since we can probably improve our programs to handle such documents. However, for documents with relatively high noise rates, it may be better to forgo sentence extraction and instead favor extraction of keywords or noun phrases. It may also be desirable to attempt to correct the noise in the extracted keywords or phrases. There has been past work on correcting spelling mistakes and errors in OCR output; these techniques would be useful in noisy documents.

The quality of text processing is directly tied to the level of noise in a document. It is not seriously impacted in the presence of minor errors, but as errors increase, results may range from being difficult to read to incomprehensible. In this context, it would be useful to develop methods for assessing document noise levels without having access to the ground-truth. Such measurements could be incorporated into text processing algorithms for the purpose of avoiding problematic regions, thereby improving the overall readability. Past work on attempting to quantify document image quality for predicting OCR accuracy [BKN95, CHK99, GS96] addresses a related problem, but one which exhibits some significant differences. One possibility would be to establish a robust index that measures whether a given section of text is processable.

## References

- [BKN95] Luis R. Blando, Junichi Kanai, and Thomas A. Nartker. Prediction of OCR accuracy using simple image features. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 319–322, Montréal, Canada, August 1995.
- [CHK99] Michael Cannon, Judith Hochberg, and Patrick Kelly. Quality assessment and restoration of typewritten document images. Technical Report LA-UR 99-1233, Los Alamos National Laboratory, 1999.

- [ELS94] Jeffrey Esakov, Daniel P. Lopresti, and Jonathan S. Sandberg. Classification and distribution of optical character recognition errors. In *Proceedings of Document Recognition I (IS&T/SPIE Electronic Imaging)*, volume 2181, pages 204–216, San Jose, CA, February 1994.
- [ELSZ94] Jeffrey Esakov, Daniel P. Lopresti, Jonathan S. Sandberg, and Jiangying 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.
- [GR00] Y. Gotoh and S. Renals. Sentence boundary detection in broadcast speech transcripts. In *Proceedings of ISCA Tutorial and Research Workshop ASR-2000*, Paris, France, 2000.
- [GS96] Venu Govindaraju and Sargur N. Srihari. Assessment of image quality to predict readability of documents. In *Proceedings of Document Recognition III (IS&T/SPIE Electronic Imaging)*, volume 2660, pages 333–342, San Jose, CA, January 1996.
- [HF01] C. Hori and S. Furui. Advances in automatic speech summarization. In *Proceedings of the 7th European Conference on Speech Communication and Technology*, pages 1771–1774, Aalborg, Denmark, 2001.
- [JLS03] Hongyan Jing, Daniel Lopresti, and Chilin Shih. Summarizing noisy documents. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 111–119, April 2003.
- [Lop01] Daniel P. Lopresti. A comparison of text-based methods for detecting duplication in scanned document databases. *Information Retrieval*, 4(2):153–173, July 2001.
- [Mac95] Robert MacIntyre. Penn Treebank tokenizer (sed script source code), 1995. <http://www.cis.upenn.edu/treebank/tokenizer.sed>.
- [MBS<sup>+</sup>00] D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel. Named entity extraction from noisy input: Speech and OCR. In *Proceedings of the 6th Applied Natural Language Processing Conference*, pages 316–324, Seattle, WA, 2000.
- [NW70] 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.
- [PO01] D. D. Palmer and M. Ostendorf. Improving information extraction by modeling errors in speech recognizer output. In J. Allan, editor, *Proceedings of the First International Conference on Human Language Technology Research*, 2001.
- [Pro04] Project Gutenberg, September 2004. <http://www.gutenberg.net/>.

- [Rat96] Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*, May 1996. <ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz>.
- [RR97] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC, March-April 1997. <ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz>.
- [S<sup>+</sup>04] Joerg Schulenburg et al. GOCR (open source software), September 2004. <http://jocr.sourceforge.net/index.html>.
- [TBC96a] Kazem Taghva, Julie Borsack, and Allen Condit. Effects of OCR errors on ranking and feedback using the vector space model. *Information Processing and Management*, 32(3):317–327, 1996.
- [TBC96b] Kazem Taghva, Julie Borsack, and Allen Condit. Evaluation of model-based retrieval effectiveness with OCR text. *ACM Transactions on Information Systems*, 14:64–93, January 1996.
- [WF74] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21:168–173, 1974.

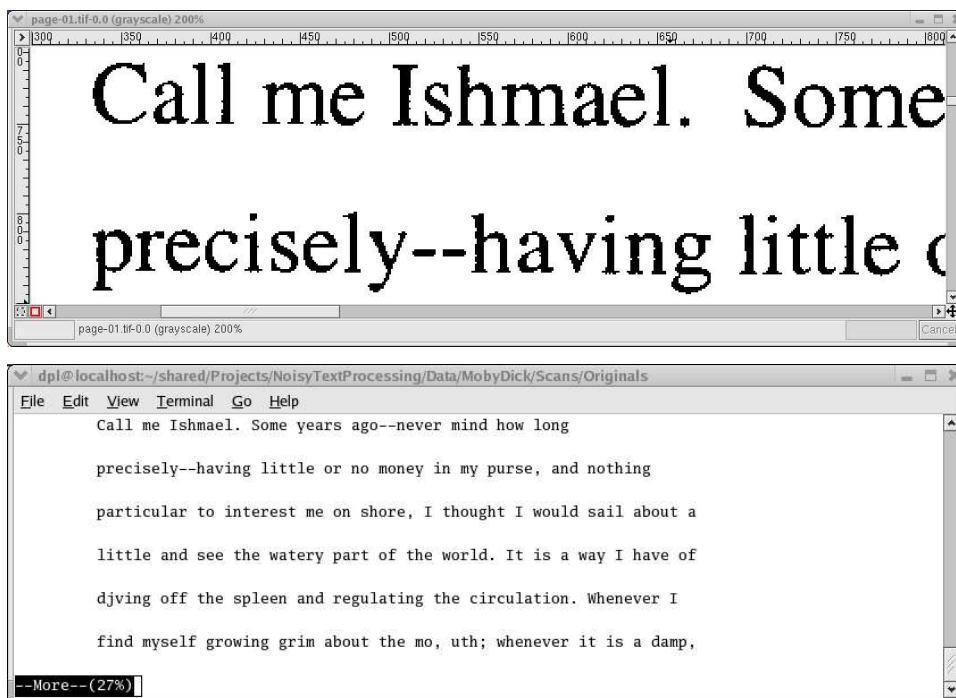


Figure 3: Image fragment and OCR output for clean input document.

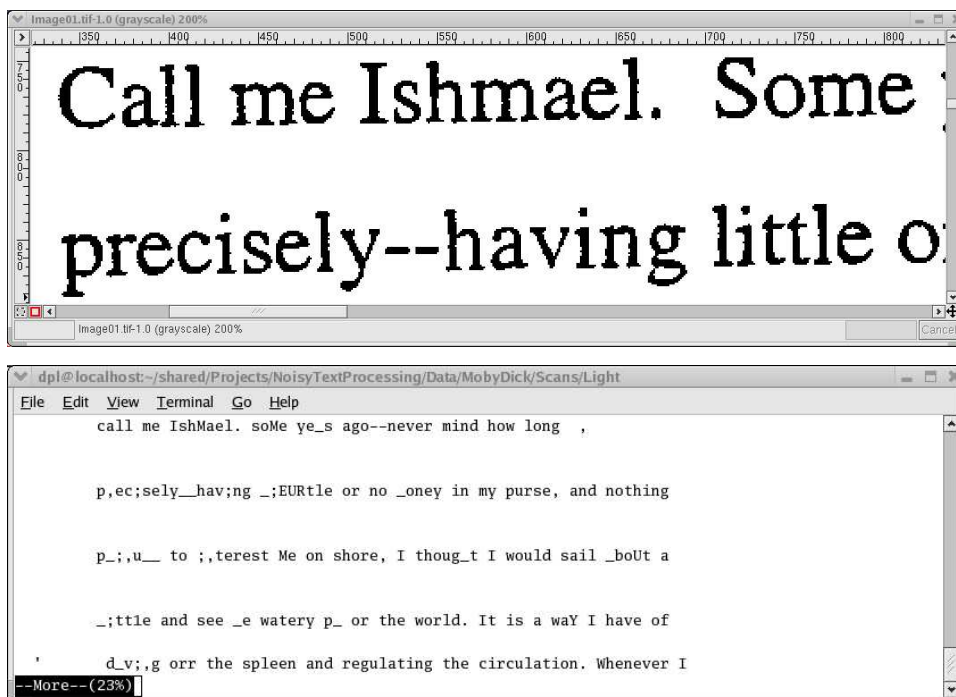


Figure 4: Image fragment and OCR output for light input document.

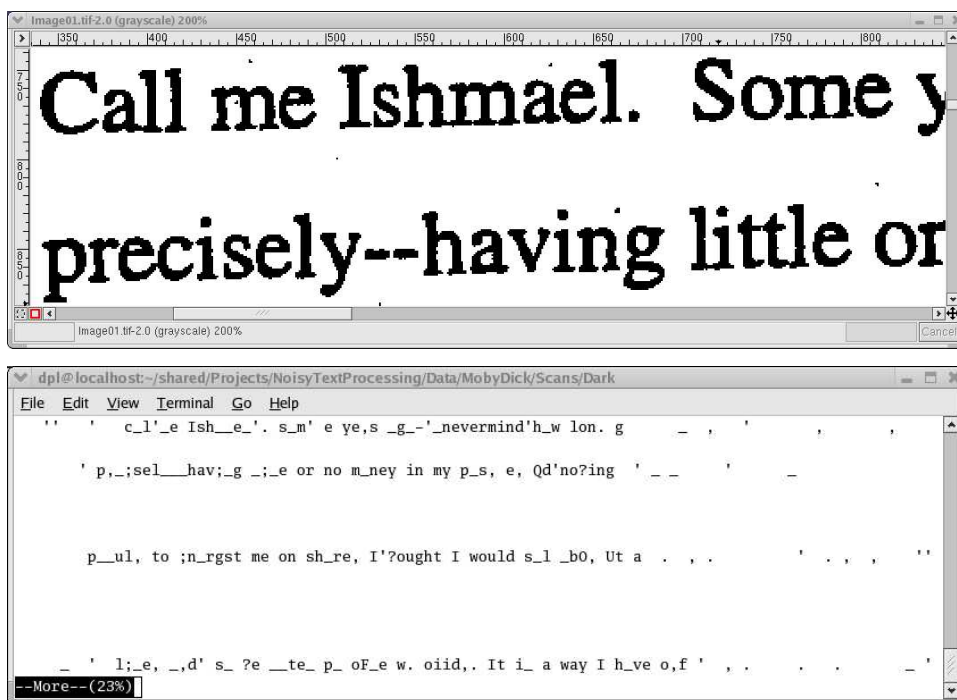


Figure 5: Image fragment and OCR output for dark input document.

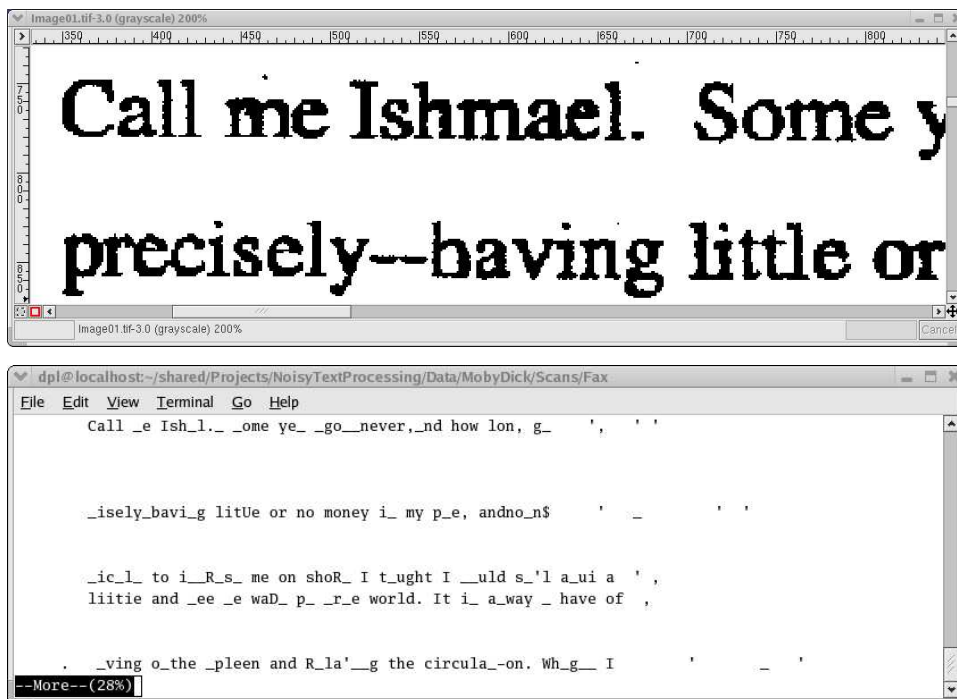


Figure 6: Image fragment and OCR output for fax input document.