Measuring the Impact of Character Recognition Errors on Downstream Text Analysis *

Daniel Lopresti

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

ABSTRACT

Noise presents a serious challenge in optical character recognition, as well as in the downstream applications that make use of its outputs as inputs. In this paper, we describe a paradigm for measuring the impact of recognition errors on the stages of a standard text analysis pipeline: sentence boundary detection, tokenization, and part-of-speech tagging. Employing a hierarchical methodology based on approximate string matching for classifying errors, their cascading effects as they travel through the pipeline are isolated and analyzed. We present experimental results based on injecting single errors into a large corpus of test documents to study their varying impacts depending on the nature of the error and the character(s) involved. While most such errors are found to be localized, in the worst case some can have an amplifying effect that extends well beyond the site of the original error, thereby degrading the performance of the end-to-end system.

Keywords: Optical character recognition, text analysis, natural language processing, error classification

1. INTRODUCTION

Real-world data is noisy. The outputs from optical character recognition (OCR) and automatic speech recognition (ASR) systems, for example, often contain errors. The more highly degraded the input, the greater the error rate. Since such systems often form the first stage in a pipeline where later stages are designed to support sophisticated information extraction and exploitation applications, an important question to be asked concerns the effects of recognition errors on downstream text analysis routines. Are all recognition errors equal in their impact, or are some worse than others? Can the performance of each stage be optimized in isolation, or must the end-to-end system be considered? The answers to these questions should influence the way we design and build document analysis systems.

Researchers have already begun studying problems relating to processing text data from noisy sources. To date, this work has focused predominately on the kinds of 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.[1] Palmer and Ostendorf describe an approach for improving named entity extraction by explicitly modeling speech recognition errors through the use of statistics annotated with confidence scores.[2] The recent *Workshop on Analytics for Noisy Unstructured Text Data*,[3] held in conjunction with the *Twentieth International Joint Conference on Artificial Intelligence*, featured papers examining the problem of noise from a variety of perspectives, largely focusing on the ambiguities that are inherent in informal written and spoken language.

There has been 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,[4,5] 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,[6] 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.

* To be presented at *Document Recognition and Retrieval XV*, San Jose, CA, January 2008.
E-mail: lopresti@cse.lehigh.edu, Telephone: (610) 758-5782

A 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.[7] 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.

In a later paper,[8] we turned to the specific question of performance evaluation for text analysis pipelines, proposing a paradigm based the hierarchical application of approximate string matching techniques. This flexible yet mathematically rigorous approach both quantifies the performance of a given processing stage as well as identifies explicitly the errors it has made. Also presented were the results of a pilot study where a small set of documents were OCR'ed and then piped through procedures for performing sentence boundary detection, tokenization, and part-of-speech tagging, demonstrating the utility of the approach.

In the present paper, we employ the same evaluation paradigm as in this previous work, but focus on the intriguing question of whether certain kinds of character recognition errors are worse than others for downstream text analysis. We present experimental results based on injecting single errors into a large corpus of test documents to study their varying impacts based on the nature of the error and the character(s) involved. In the worst case, we find there can be an "error cascade" with amplifying effects that extend well beyond the site of the original error. We conclude by outlining some topics for future research to delve more deeply into this issue.

2. STAGES IN TEXT ANALYSIS

In this section, we describe in general terms the stages that are common to many text analysis systems, and then list the specific packages we use. The stages, in order, are typically: (1) optical character recognition, (2) sentence boundary detection, (3) tokenization, and (4) part-of-speech tagging. These basic procedures are of interest because they form the basis for more sophisticated natural language applications, including named entity identification and extraction, topic detection and clustering, and summarization.

In selecting implementations of the above stages, we choose to employ freely available open source software rather than proprietary, commercial solutions. From the standpoint of our evaluation, all we require is that the behavior be representative, not necessarily "best-in-class." For sufficiently noisy inputs, the same methodology and conclusions are likely to apply to other approaches to performing these tasks, 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 deteriorates in the case of degraded documents, complex layouts, and/or unusual fonts. 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 *Tesseract* package.[9] Since we are presenting it with relatively simple text layouts, having to contend with complex documents is not a concern in our experiments. The performance of *Tesseract* 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. Figure 1 shows a portion of a faxed page used in our studies, while Figure 2 shows the OCR output from *Tesseract*. Note that the darkening and smearing of character shapes and the speckle noise leads to a number of substitution errors (*e.g.*, `m` → `ni`, `e` → `c`, `o` → `p`, `n` → `b`) and insertion errors (*e.g.*, → `‘`, → `.`).

In this controlled experiment, we do not use the output from OCR directly. Rather, we perform an error analysis that involves OCR'ing real scanned pages to create a set of confusion matrices for "injecting" OCR errors into otherwise clean text. In this way, we can measure the impact of a single error without having to worry about the confounding effects of other, neighboring errors.

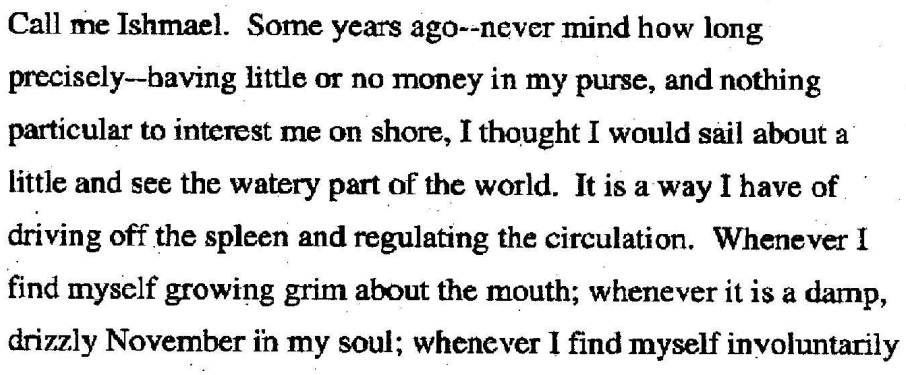

Figure 1. Example of a portion of a fax image.

```
Call nic Ishmael.  Some years ago--never mind how long-
precisely-having little or no money in my purse, and nothing
particular to interest me onshore, I thought I would sail about a‘
little and see the watery part pf the world. It is away I have of
driving off the spleen and regulating the circulation.  Whenever I
find myself growing grim about the mputh; whenever it is a damp,
drizzly November ib my soul; whenever I End myself involuntarily
```

Figure 2. OCR output for the image from Figure 1.

2.2 Sentence Boundary Detection

Procedures for sentence boundary detection use a variety of syntactic and 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). The sentence boundary detector we used in our test is the MXTERMINATOR package by Reynar and Ratnaparkhi.[10] An example of its output for a "clean" (error-free) text fragment consisting of two sentences is shown in Figure 3(b).[†]

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.[11] As noted in the documentation for that system, its operation can be summarized as: (1) most punctuation is split from adjoining words, (2) double quotes are changed to doubled single forward- and backward-quotes, and (3) verb contractions and the Anglo-Saxon genitive of nouns are split into their component morphemes, and each morpheme is tagged separately. Sample output for the tokenization routine is shown in Figure 3(c).

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,[12] which produced a total of 42 different part-of-speech tags for our test inputs. Its output using the input produced by the previous two stages is given in Figure 3(d).

At this point, it is also good for us to note that the evaluations we conduct in this work are *relativistic*. That is, there is no universal ground-truth, but rather we compare the performance of the various text analysis stages on clean and noisy inputs. An "error" is considered to have occurred when the two sets of results differ. There may already in fact be errors present, even for clean inputs. For example, in Figure 3(d), the first two words in the noun phrase "fitness assessment programs" should be labeled as adjectives (JJ), not as nouns (NN).

[†]Due to line-length limitations in the figures in this paper, we indicate continuations in the case of longer sentences through the use of the backslash character.

(a) Input text.

```
National Health Enhancement Systems Inc said it is offering a new health
evaluation system to its line of fitness assessment programs. The company
said the program, called The Health Test, will be available in 60 days.
```

(b) Sentence boundary detection output.

```
National Health Enhancement Systems Inc said it is offering a new health \
   evaluation system to its line of fitness assessment programs.
The company said the program, called The Health Test, will be available in 60 days.
```

(c) Tokenization output.

```
National Health Enhancement Systems Inc said it is offering a new health \
   evaluation system to its line of fitness assessment programs .
The company said the program , called The Health Test , will be available in 60 days .
```

(d) Part-of-speech tagging output.

```
National_NNP Health_NNP Enhancement_NNP Systems_NNPS Inc_NNP said_VBD it_PRP \
   is_VBZ offering_VBG a_DT new_JJ health_NN evaluation_NN system_NN to_TO its_PRP$ \
   line_NN of_IN fitness_NN assessment_NN programs_NNS ._.
The_DT company_NN said_VBD the_DT program_NN ,_, called_VBD The_DT Health_NNP \
   Test_NNP ,_, will_MD be_VB available_JJ in_IN 60_CD days_NNS ._.
```

Figure 3. Example output from the text analysis stages.

3. PERFORMANCE EVALUATION

Performance evaluation for text analysis of noisy inputs presents some serious challenges. The approach we described in our earlier paper makes use of approximate string matching to align two linear streams of text, one representing OCR output and the other representing the ground-truth.[8]

Due to the markup conventions employed by sentence boundary detection, tokenization, and part-of-speech tagging, as just described, this task is significantly more complex than the basic alignment problem used for assessing raw OCR accuracy.[13,14] Still, we can build on the same paradigm, employing an optimization framework that likewise can be solved using dynamic programming.[15] We begin by letting $S = s_1 s_2 \ldots s_m$ be the source document (the ground-truth), $T = t_1 t_2 \ldots t_n$ be the target document (the OCR output), and defining $dist1_{i,j}$ 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. This information is used in analyzing the performance of the procedure under study.

To generalize these ideas to later stages of text processing, 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.

In the paradigm we have developed, we adopt a three level hierarchy.[8] 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 and 2).

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

$$dist2_{i,j} = \min \begin{cases} dist2_{i-1,j} \ + \ c2_{del}(s_i) \\ dist2_{i,j-1} \ + \ c2_{ins}(t_j) \\ \min_{1 \le k' \le k,\ 1 \le l' \le l} [\ dist2_{i-k',j-l'} \ + \\ \qquad c2_{sub_{k:l}}(s_{i-k'+1 \ldots i}, t_{j-l'+1 \ldots j})] \end{cases} \tag{3}$$

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, which can be naturally 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:

$$dist3_{i,j} = \min \begin{cases} dist3_{i-1,j} \ + \ c3_{del}(s_i) \\ dist3_{i,j-1} \ + \ c3_{ins}(t_j) \\ \min_{1 \le k' \le k,\ 1 \le l' \le l} [\ dist3_{i-k',j-l'} \ + \\ \qquad c3_{sub_{k:l}}(s_{i-k'+1 \ldots i}, t_{j-l'+1 \ldots j})] \end{cases} \tag{4}$$

with costs defined in terms of the second-level computation.

By executing this hierarchical dynamic programming from the top down, given an input page for the OCR results as processed through the text analysis pipeline and another page for the corresponding 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.

From a practical standpoint, this optimization process can require a substantial amount of CPU time depending on the length of the input documents (we have observed runtimes of several minutes for pages containing $\pm 1,000$ characters). There are, however, well-known techniques for speeding up dynamic programming (*e.g.*, so-called "beam search") which have little or no effect on the optimality of the results for the cases of interest.

4. EXPERIMENTAL RESULTS

To build our OCR confusion matrices, we took 10 pages of text from the opening chapters of the Project Gutenberg edition of Herman Melville's classic novel *Moby-Dick*,[16] 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 *Tesseract* OCR package. Examples of a small region of a scanned page image and the associated OCR output from this dataset were shown in Figures 1 and 2.

Basic OCR accuracy can be judged using a single level of dynamic programming, *i.e.*, Equations 1 and 2, as described elsewhere.[13,14] These results for the four datasets are presented in Table 1. As in the information retrieval domain, precision and recall are used here to reflect two different aspects of system performance. The

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

	All Symbols			Punctuation			Whitespace		
	Prec.	Recall	Overall	Prec.	Recall	Overall	Prec.	Recall	Overall
Clean	0.995	0.996	0.997	0.975	0.995	0.987	0.995	0.999	0.997
Light	0.986	0.990	0.992	0.939	0.988	0.967	0.989	0.998	0.994
Dark	0.934	0.965	0.963	0.718	0.955	0.823	0.922	0.979	0.948
Fax	0.954	0.974	0.974	0.778	0.923	0.859	0.938	0.983	0.960

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 baseline OCR accuracy is quite high, but performance deteriorates 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. Punctuation symbols in particular are badly impacted, with a large number of false alarms (low precision), especially in the cases of the Dark and Fax datasets where fewer than 80% of the reports are true. This phenomenon has serious implications for sentence boundary detection and later stages of text processing.

As noted earlier, we used the error characteristics from this real OCR output to build confusion matrices for our controlled study. Each observed error was injected, one at a time, into a randomly selected document from the well-known Reuters-21578 news corpus.[17] All-in-all, there were 33 unique deletion events, 62 unique insertion events, and 276 unique substitution events. However, since most of these errors occurred more than once, 1,339 error patterns were tested in total. As might be anticipated, the most frequently occurring deletion and insertion events involved space and punctuation characters, while the substitution errors were more uniformly distributed.

We then ran sentence boundary detection, tokenization, and part-of-speech tagging on both the original (ground-truth) news story and the version with the simulated OCR error, comparing the results using the paradigm described in the previous section. This allowed us to both quantify performance as well as to determine the optimal alignments between sequences and hence identify the actual errors that had arisen.

An example of a relatively straightforward alignment produced by our evaluation procedure is shown in Figure 4. This displays the effect of a single-character substitution error (`i` being misrecognized as `;`). The result is three tokens where before there was only one. Not unexpectedly, two of the three tokens have inappropriate part-of-speech labels. In this instance, the OCR error impacts two text analysis stages and is, for the most part, localized; other errors can have effects that cascade through the pipeline, becoming amplified at each stage.

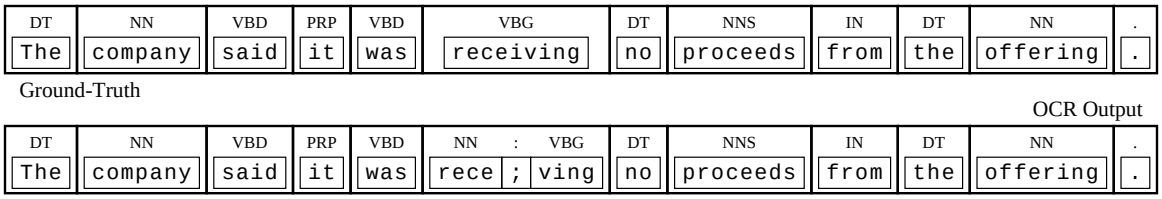

Figure 4. Example of an alignment displaying impact of a single substitution error.

A tabulation of the dynamic programming results for the three text processing stages appears in Table 2, broken down in terms of error type (insertion, deletion, or substitution), and in Table 3, broken down in terms of character class for the error character(s). Here we see that a lone character recognition error can have, at times, a relatively large impact on one or more of the downstream text analysis stages. Sentence boundary detection appears particularly susceptible in the worst case. Such errors may induce additional mistakes at later stages in the pipeline, a motivating question in our work.

In considering the results reported in Tables 2 and 3, it must be remarked the standard document-level analysis reflected here is not ideal for the question at hand. We are injecting just one error in a document that contains hundreds or thousands of characters. Hence, it is more instructive to focus directly on the errors

Table 2. Document-level accuracy for a single injected OCR error (by error type).

	Sentence Boundaries			Tokenization			Part-of-Speech Tagging		
	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.
All Errors	0.997	0.667	1.000	0.999	0.949	1.000	0.994	0.904	1.000
Deletion	0.999	0.889	1.000	0.999	0.984	1.000	0.995	0.950	1.000
Insertion	0.994	0.667	1.000	0.999	0.977	1.000	0.992	0.909	1.000
Substitution	0.998	0.667	1.000	0.999	0.949	1.000	0.995	0.904	1.000

Table 3. Document-level accuracy for a single injected OCR error (by character class of error character(s)).

	Sentence Boundaries			Tokenization			Part-of-Speech Tagging		
	Ave.	Min.	Max.	Ave.	Min.	Max.	Ave.	Min.	Max.
All Symbols	0.997	0.667	1.000	0.999	0.949	1.000	0.994	0.904	1.000
Punctuation	0.983	0.667	1.000	0.997	0.977	1.000	0.990	0.931	1.000
Whitespace	0.995	0.857	1.000	0.994	0.963	1.000	0.986	0.912	1.000

themselves and the effects they induce. In Table 4, we track the impact of a single OCR error on a per-stage basis by the number of errors it causes in downstream text analysis routines. While always altering the input text, in the best case, of course, such an error might result in no mistakes in sentence boundary detection, tokenization, or part-of-speech tagging: those procedures could be agnostic to the error in question. Here, however, we can see that nearly 35% of the injected OCR errors cause at least one change in a part-of-speech tag. Moreover, almost 16% result in a different number of tokens than are found in the original text input, and somewhat over 2% change sentence boundaries.

Table 4. Per-stage impact of a single OCR error event over 1,339 instances.

Total Errors	Sentence Boundaries		Tokenization		Part-of-Speech Tagging
in Stage	Missed	Spurious	Missed	Spurious	Incorrect
1	8	21	94	105	379
2	–	–	2	2	73
3	–	–	–	5	4
4	–	–	5	–	4
5	–	–	–	1	4
9	–	–	–	–	1
18	–	–	–	–	1
Total (Pct)	8 (0.6%)	21 (1.6%)	101 (7.5%)	113 (8.4%)	466 (34.8%)

We are also interested in examining the cascade effect: the tendency for a single error in an earlier stage to give rise to additional errors in later stages. On occasion, the number of induced errors may actually grow as the document passes through the text analysis pipeline. For example, in Figure 5, we see that one simple error, substituting a period (`.`) for a lower-case `l` ("el") and thereby changing the word "`will`" into "`wil.`", induces:

1. a spurious sentence boundary after what should have been the word "`will`";
2. a spurious token, since the new period is segmented from the characters that precede it;
3. a spurious part-of-speech tag, since the new period token gets tagged individually;
4. two changes to the part-of-speech tagging for surrounding words: "`will`" goes from `MD` to `NN` in error word "`wil`", and "`use`" goes from `VB` to `NN`.

Errors that arise in later stages of processing may be due to the original OCR error, or to an error it induced in an earlier pipeline stage. Whatever the cause, this error cascade is an important artifact of pipelined text

(a) Original text.

```
The plant will use a chemical process called leaching to
extract the residual gold, which could not otherwise be
economically recovered.
```

(b) Text with a single injected OCR error (substitution `l` → `.`).

```
The plant wil. use a chemical process called leaching to
extract the residual gold, which could not otherwise be
economically recovered.
```

(c) Text analysis output for original text.

```
The_DT plant_NN will_MD use_VB a_DT chemical_NN process_NN called_VBD leaching_VBG to_TO \
  extract_VB the_DT residual_JJ gold_NN ,_, which_WDT could_MD not_RB otherwise_RB be_VB \
  economically_RB recovered_VBN ._.
```

(d) Text analysis output for text with injected OCR error.

```
The_DT plant_NN wil_NN ._.
use_NN a_DT chemical_NN process_NN called_VBD leaching_VBG to_TO \
  extract_VB the_DT residual_JJ gold_NN ,_, which_WDT could_MD not_RB otherwise_RB be_VB \
  economically_RB recovered_VBN ._.
```

Figure 5. Example where a single OCR error induces text analysis errors at each subsequent stage.

analysis systems. In Table 5, we present one view of this cascade effect. Each "syndrome" vector groups the classes of errors that occurred together in a single test document. Over 92% of the time, an error in sentence boundary detection is indicative of additional errors in at least one of the two later stages of the pipeline, and 80% of the time it is indicative of errors in both later stages. Likewise, 60% of the time, an error in tokenization is indicative of at least one incorrect part-of-speech tag as well. Nearly 30% of the time, the number of errors increases monotonically from one stage to the next.[‡]

5. CONCLUSIONS

In this paper, we considered a text analysis pipeline consisting of four stages: optical character recognition, sentence boundary detection, tokenization, and part-of-speech tagging. Using a formal algorithmic model for evaluating the performance of multi-stage processes, we presented experimental results examining the impact of representative OCR errors on later stages in the pipeline. While most such errors are localized, in the worst case some have an amplifying effect that extends well beyond the site of the original error, thereby degrading the performance of the end-to-end system. Studies such as this provide a basis for the development of more robust text analysis techniques, as well as guidance for tuning OCR systems to achieve optimal performance when embedded in larger applications.

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. Similarly, the majority of errors that occurred in our study are part-of-speech tagging errors which, while they arise in the final stage here, would feed into additional text processing routines in a real system, contributing to a further error cascade. One obvious approach for attempting to address this issue would be to retrain existing systems on such documents to make them more tolerant of noise. This line of attack would be analogous to techniques now being developed to improve NLP performance on informal and/or ungrammatical text.[3,18] However, this is likely to be effective only when noise

[‡]Although not tabulated separately here, it should be clear that missed/spurious tokenizations automatically qualify as part-of-speech tagging errors as well.

Table 5. Induced error syndrome vectors over 1,339 instances.

Number of Occurrences	Sentence Boundaries Missed	Sentence Boundaries Spurious	Tokenization Missed	Tokenization Spurious	Part-of-Speech Tagging Incorrect
18 total errors across all stages					
1	0	0	0	0	18
10 total errors across all stages					
1	0	1	4	5	0
1	0	0	0	1	9
9 total errors across all stages					
4	0	1	4	3	1
6 total errors across all stages					
1	0	1	2	3	0
5 total errors across all stages					
1	1	0	0	0	4
4	0	0	0	0	5
4 total errors across all stages					
1	0	1	0	1	2
1	0	0	0	1	3
3	0	0	0	0	4
3 total errors across all stages					
1	1	0	1	0	1
2	0	1	0	1	1
2	0	1	0	0	2
1	0	0	2	0	1
7	0	0	1	0	2
1	0	0	0	2	1
7	0	0	0	1	2
3	0	0	0	0	3
2 total errors across all stages					
2	1	0	1	0	0
4	1	0	0	0	1
1	0	1	1	0	0
4	0	1	0	1	0
3	0	1	0	0	1
42	0	0	1	0	1
1	0	0	0	2	0
29	0	0	0	1	1
56	0	0	0	0	2
1 total error across all stages					
2	0	1	0	0	0
41	0	0	1	0	0
60	0	0	0	1	0
292	0	0	0	0	1

levels are relatively low. Significant work needs to be done to develop robust methods that can handle documents with high noise levels.

In the context of document summarization, a potential downstream application, we have previously noted that the quality of text analysis is directly tied to the level of noise in a document.[7] Summaries are not seriously impacted in the presence of minor errors, but as errors increase, the results may range from being difficult to read, to incomprehensible. Here it would be useful to develop methods for assessing noise levels in an input image without requiring access to ground-truth. Such measurements could be incorporated into text analysis algorithms for the purpose of segmenting out problematic regions of the page for special processing (or even avoiding them entirely), thereby improving overall readability. Past work on attempting to quantify document

image quality for predicting OCR accuracy[19–21] addresses a related problem, but one which exhibits some notable differences. One possibility would be to establish a robust index that measures whether a given section of text is processable.

We close by noting that the evaluation paradigm we have described is general and not limited to cases of isolated OCR errors; the experiments described in this paper are admittedly artificial in that regard. Rarely do real-world documents of interest contain just a single error. The work we have presented does provide a lower-bound, however, since it is highly unlikely that additional errors will improve later-stage performance. This raises interesting questions for future research concerning the interactions between multiple OCR errors that might occur in close proximity, as well as higher-level document analysis errors that can impact larger regions of the page. Are such errors further amplified downstream? Is the cumulative effect more additive or multiplicative? The answers to questions such as these will prove important as we seek to build more sophisticated systems capable of handling real-world document processing tasks for inputs that range both widely in content and quality.

REFERENCES

1. Y. Gotoh and S. Renals, "Sentence boundary detection in broadcast speech transcripts," in *Proceedomgs of ISCA Tutorial and Research Workshop ASR-2000*, (Paris, France), 2000.
2. D. D. Palmer and M. Ostendorf, "Improving information extraction by modeling errors in speech recognizer output," in *Proceedings of the First International Conference on Human Language Technology Research*, J. Allan, ed., 2001.
3. *Workshop on Analytics for Noisy Unstructured Text Data*, Hyderabad, India, January 2007. http://research.ihost.com/and2007/.
4. K. Taghva, J. Borsack, and A. Condit, "Effects of OCR errors on ranking and feedback using the vector space model," *Information Processing and Management* **32**(3), pp. 317–327, 1996.
5. K. Taghva, J. Borsack, and A. Condit, "Evaluation of model-based retrieval effectiveness with OCR text," *ACM Transactions on Information Systems* **14**, pp. 64–93, January 1996.
6. 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*, pp. 316–324, (Seattle, WA), 2000.
7. H. Jing, D. Lopresti, and C. Shih, "Summarizing noisy documents," in *Proceedings of the Symposium on Document Image Understanding Technology*, pp. 111–119, April 2003.
8. D. Lopresti, "Performance evaluation for text processing of noisy inputs," in *Proceedings of the 20th Annual ACM Symposium on Applied Computing (Document Engineering Track)*, pp. 759–763, (Santa Fe, NM), March 2005.
9. "Tesseract open source OCR engine," November 2007. http://sourceforge.net/projects/tesseract-ocr.
10. J. C. Reynar and A. 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.
11. R. MacIntyre, "Penn Treebank tokenizer (sed script source code)," 1995. http://www.cis.upenn.edu/ treebank/tokenizer.sed.
12. A. 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.
13. J. Esakov, D. P. Lopresti, and J. S. Sandberg, "Classification and distribution of optical character recognition errors," in *Proceedings of Document Recognition I (IS&T/SPIE Electronic Imaging)*, **2181**, pp. 204–216, (San Jose, CA), February 1994.
14. 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*, pp. 401–412, April 1994.
15. R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," *Journal of the Association for Computing Machinery* **21**, pp. 168–173, 1974.

16. "Project Gutenberg," November 2007. http://www.gutenberg.net/.
17. D. D. Lewis, "Reuters-21578 test collection, distribution 1.0," November 2007. http://www.daviddlewis.com/resources/testcollections/reuters21578/.
18. J. Foster, "Treebanks gone bad: Generating a treebank of ungrammatical English," in *Proceedings of the Workshop on Analytics for Noisy Unstructured Text Data*, (Hyderabad, India), January 2007. http://research.ihost.com/and2007/cd/Proceedings_files/p39.pdf.
19. L. R. Blando, J. Kanai, and T. A. Nartker, "Prediction of OCR accuracy using simple image features," in *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 319–322, (Montréal, Canada), August 1995.
20. M. Cannon, J. Hochberg, and P. Kelly, "Quality assessment and restoration of typewritten document images," Tech. Rep. LA-UR 99-1233, Los Alamos National Laboratory, 1999.
21. V. Govindaraju and S. N. Srihari, "Assessment of image quality to predict readability of documents," in *Proceedings of Document Recognition III (IS&T/SPIE Electronic Imaging)*, **2660**, pp. 333–342, (San Jose, CA), January 1996.