A Statistical Approach to Automatic OCR Error Correction in Context

Xiang Tong and David A. Evans
Laboratory for Computational Linguistics
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.
{*tong, dae*}*@lcl.cmu.edu*

Abstract

This paper describes an automatic, context-sensitive, word-error correction system based on statistical language modeling (SLM) as applied to optical character recognition (OCR) post-processing. The system exploits information from multiple sources, including letter n-grams, character confusion probabilities, and word-bigram probabilities. Letter n-grams are used to index the words in the lexicon. Given a sentence to be corrected, the system decomposes each string in the sentence into letter n-grams and retrieves word candidates from the lexicon by comparing string n-grams with lexicon-entry n-grams. The retrieved candidates are ranked by the conditional probability of matches with the string, given character confusion probabilities. Finally, the word-bigram model and Viterbi algorithm are used to determine the best scoring word sequence for the sentence. The system can correct non-word errors as well as real-word errors and achieves a 60.2% error reduction rate for real OCR text. In addition, the system can learn the character confusion probabilities for a specific OCR environment and use them in self-calibration to achieve better performance.

1 Introduction

Word errors present problems for various text- or speech-based applications such as optical character recognition (OCR) and voice-input computer interfaces. In particular, though current OCR technology is quite refined and robust, sources such as old books, poor-quality (nth-generation) photocopies, and faxes can still be difficult to process and may cause many OCR errors. For OCR to be truly useful in a wide range of applications, such as office automation and information retrieval systems, OCR reliability must be improved. A method for the automatic correction of OCR errors would be clearly beneficial.

Essentially, there are two types of word errors: *non-word errors* and *real-word errors*. A non-word error occurs when a word in a source text is interpreted (under OCR) as a string that does not correspond to any valid word in a given word list or dictionary. A real-word error occurs when a source-text word is interpreted as a string that actually does occur in the dictionary, but is not identical with the source-text word. For example, if the source text "John found the man" is rendered as "John fornd he man" by an OCR device, then "fornd" is a non-word error and "he" is a real-word error. In general, non-word errors will never correspond to any dictionary entries and

will include wildly incorrect strings (such as "#~~&&") as well as misrecognized alpha-numeric sequences (such as "BN234" for "8N234"). However, some non-word errors might become real-word errors if the size of the word list or dictionary increases. (For example, the word "ruel"[1] might count as a non-word error for the source-text word "rut" if a small dictionary is used for reference, but count as a real-word error if an unabridged dictionary is used.) While non-word errors might be corrected without considering the context in which the error occurs, a real-word error can only be corrected by taking context into account.

The problems of word-error detection and correction have been studied for several decades. A good survey in this area can be found in [Kukich 1992]. Most traditional word-correction techniques concentrate on non-word error correction and do not consider the context in which the error appears.

Recently, statistical language models (SLMs) and feature-based methods have been used for context-sensitive spelling-error correction. For example, Atwell and Elliittm [1987] have used a part-of-speech (POS) tagging method to detect the real-word errors in text. Mays and colleagues [1991] have exploited word trigrams to detect and correct both the non-word and real-word errors that were artificially generated from 100 sentences. Church and Gale [1991] have used a Bayesian classifier method to improve the performance for non-word error correction. Golding [1995] has applied a hybrid Bayesian method for real-word error correction and Golding and Schabes [1996] have combined a POS trigram and Bayesian methods for the same purpose.

The goal of the work described here is to investigate the effectiveness and efficiency of SLM-based methods applied to the problem of OCR error correction. Since POS-based methods are not effective in distinguishing among candidates with the same POS tags and since methods based on word-trigram models involve extensive training data and require that huge word-trigram tables be available at run time, we used a word-bigram SLM as the first step in our investigation.

In this paper, we describe a system that uses a word-bigram SLM technique to correct OCR errors. The system takes advantage of information from multiple sources, including letter n-grams, character confusion probabilities, and word bigram probabilities, to effect context-based word error correction. It can correct non-word as well as real-word errors. In addition, the system can learn the character confusion probability table for a specific OCR environment and use it to achieve better performance.

2 The Approach

2.1 Problem Statement

The problem of context-based OCR word-error correction can be stated as follows:

Let $L = \{w_1, w_2, ..., w_m\}$ be the set of all the words in a given lexicon. For an input sentence, $S = s_1, ..., s_n$, produced as the output of an OCR device, where $s_1, ..., s_n$ are character strings separated by spaces, find the best word sequence, $\hat{W} = w_1, w_2, ..., w_n$, for $w_i \in L$, that maximizes the probability $pr(W|S)$:

$$\hat{W} = argmax_W(pr(W|S)) \quad (1)$$

[1] "Ruel" is an obscure French-derivative word meaning the space between a bed and the wall.

Using Bayes' formula, we can rewrite 1 as:

$$\begin{aligned} & argmax_W(pr(W|S)) \\ = \; & argmax_W(\frac{pr(W) * pr(S|W)}{pr(S)}) \\ = \; & argmax_W(pr(W) * pr(S|W)) \end{aligned} \tag{2}$$

The probability $pr(W)$ is given by the language model and can be decomposed as:

$$pr(W) = \prod_{i=1}^{n} pr(w_i|w_{1,i-1}) \tag{3}$$

where $pr(w_i|w_{1,i-1})$ is the probability that the word w_i appears given that $w_1, w_2, \ldots, w_{i-1}$ appeared previously.

In a word-bigram language model, we assume that the probability that a word w_i will appear is affected only by the immediately preceding word. Thus,

$$pr(w_i|w_{1,i-1}) = pr(w_i|w_{i-1}) \tag{4}$$

and

$$pr(W) = \prod_{i=1}^{n} pr(w_i|w_{i-1}) \tag{5}$$

The conditional probability, $pr(S|W)$, reflects the channel (processing) characteristics of the OCR environment. If we assume that strings produced under OCR are independent of one another, we have the following formula:

$$pr(S|W) = \prod_{i=1}^{n} pr(s_i|w_i) \tag{6}$$

So,

$$\begin{aligned} \hat{W} &= argmax_W(pr(W) * pr(S|W)) \\ &= argmax_W(\prod_{i=1}^{n} pr(w_i|w_{i-1}) * pr(s_i|w_i)) \end{aligned} \tag{7}$$

Thus, the problem of calculating W is reduced to estimating the word-bigram probability, $pr(w_i|w_{i-1})$, and the word confusion probability, $pr(s_i|w_i)$. The word-bigram probability, $pr(w_i|w_{i-1})$, can be estimated by a *maximum likelihood* estimator (MLE):

$$pr_{ML}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

where $c(w_{i-1})$ is the number of times that w_{i-1} occurs in the text and $c(w_{i-1}, w_i)$ is the number of times that the word bigram (w_{i-1}, w_i) occurs in the text.

However, the estimatation of unseen bigrams is a problem. We use a *back-off* model similar to that described in [Dagan & Pereira 1994] to estimate the word-bigram probabilities in our system.

If we already have estimates of the probabilities $pr(w_i|w_{i-1})$ and $pr(s_i|w_i)$, the Viterbi algorithm [Charniak 1993] could be used to determine the best word sequence for the given sentence. Details of the *back-off* model and Viterbi algorithm can be found in [Dagan & Pereira 1994] and [Charniak 1993].

2.2 Estimate of Channel Probabilities and Learning of Character Confusion Table

The probability $pr(s|w)$—the conditional probability that, given a word w, it is recognized by the OCR software as the string s—can be estimated by the confusion probabilities of the characters in s if we assume that character recognition in OCR is an independent process.

We assume that an OCR string is generated from the original word by one or more of the following operations: (a) delete a character; (b) insert a character; or (c) substitute one character for another. Under such circumstances, a dynamic programming method can be used to determine the operations that maximize the conditional probability when transforming the original word to the OCR string, given a character confusion probability table.

Let $t_1, t_2 \ldots t_i$ be the first i characters of the string that is produced by the OCR process for a source word s and let $s_1, s_2 \ldots s_j$ be the first j actual characters of s. Define $pr(i|j)$ to be the conditional probability that the substring $s_{1,j}$ is recognized as substring $t_{1,i}$ by the OCR process, i.e., $pr(t_{1,i}|s_{1,j})$. The dynamic programming recurrence is given as follows:

$$pr(i|j) = max \begin{cases} pr(i-1|j) * pr(ins(t_i)) \\ pr(i|j-1) * pr(del(s_j)|s_j) \\ pr(i-1|j-1) * pr(t_i|s_j) \end{cases} \tag{8}$$

where $pr(ins(y))$ is the probability that letter y is inserted.
$pr(del(y)|y)$ is the probability that letter y is deleted.
$pr(x|y)$ is the probability that letter y is replaced by letter x.

For example, suppose that source word "flag" is recognized as "flo" by an OCR device. Formula 8 may determine that a sequence of four operations—(1) substitute "f" for "f"; (2) substitute "l" for "l"; (3) substitute "a" for "o", and (4) delete "g"—maximizes the conditional probability $pr(\text{"flo"}|\text{"flag"})$. Then the probability of "flag" being rendered as "flo" can be estimated as:

$$pr(\text{"flo"}|\text{"flag"}) = pr(\text{"f"}|\text{"f"}) * pr(\text{"l"}|\text{"l"}) * pr(\text{"o"}|\text{"a"}) * pr(del(\text{"g"})|\text{"g"})$$

This method is similar to what was described in [Wagner 1974] where the *minimum edit distance* between two strings was computed. The *minimum edit distance* is the minimum number of operations that transform the source string to the target string. Note that to effect spelling correction, we could include *character transposition* probabilities.

If we have no information about the character confusion probabilities, we can estimate them as:

$$pr(y|x) = \begin{cases} \alpha & \text{if } y = x \\ \frac{1-\alpha}{N} & \text{otherwise} \end{cases} \tag{9}$$

$$pr(del(x)|x) = pr(ins(x)) = \frac{1-\alpha}{N} \tag{10}$$

where N is the total number of printable characters.

The estimator α can be regarded as the probability that a given character is correctly recognized. Our experiments show that system performance is very sensitive to the value of α, especially for real-word error correction. For example, if α is very high, then the probability $pr(s|s)$ will be too high to be affected by subsequent processing and will not be changed. On the other hand if α is very low, some correct words may be detected as real-word errors and will be changed.

If we have both the original text and the corresponding OCR output and if we assume that the errors made by a particular OCR system are not random (but semi-deterministic), we can count the

cases of substitution, deletion, and insertion using a method similar to computing the minimum edit distance between strings [Wagner 1974] and we can estimate the probabilities using formulas similar to those in [Church & Gale 1991]:

$$pr(y|x) = num(sub(x,y))/num(x) \quad (11)$$

$$pr(del(x)) = num(del(x))/num(x) \quad (12)$$

$$pr(ins(y)) = num(ins(y))/num(<all\ letters>) \quad (13)$$

Obviously, in practice, we typically do not have the original text to compare to the OCR text or to use for correction. Moreover, as noted in [Liu *et al.* 1991], the character confusion characteristics are heavily dependent on the OCR environment, encompassing everything from the performance biases of the specific OCR software to the size of characters in the source text, fonts used, individual character types, and print quality of the text being processed. It is not feasible to train on texts to acquire character confusion probabilities for each OCR environment.

The current system employs an iterative *learning-from-correcting* technique that treats the corrected OCR text as an approximation of the original text. The system starts by assuming all characters are equally likely to be misrecognized (with some uniform, small probability) and learns the character confusion probabilities by comparing the OCR text to the corrected OCR text after each pass. Then the learned character confusion probabilities are used for the next pass processing (feedback processing). This method proves to be quite effective in improving system performance.

2.3 Generation of Word Candidates for a Given String

Ideally, each word, w, in the lexicon should be compared to a given OCR string, s, to compute the conditional probability, $pr(w|s)$. However, this approach would be computationally too expensive. Instead, the system operates in two steps, first to generate the candidates and then to specify the maximal number of candidates, N, to be considered for the correction of an OCR string.

In step 1, the system retrieves a large list of word candidates for a given string. To nominate candidates, we use a *vector space* information retrieval technique [Salton 1989]: all the words in the lexicon are indexed by letter n-grams and the (OCR) string, also parsed into letter n-grams, is treated as a query over the database of lexicon entries. In particular, all words (or OCR strings) are indexed by their letter trigrams, including the 'beginning' and 'end' spaces surrounding the string. Words of four or fewer characters are also indexed by their letter bigrams. For example:

"the" ⇒ {#th, the, he#, #t, th, he, e#}
"example" ⇒ {#ex, exa, xam, mpl, ple, le#}

A given OCR string to be corrected is represented by a vector containing its letter n-grams. Using the vector as the query, the lexicon words that are similar to the word error are retrieved, giving a large list of candidate correct forms. Candidates must share at least some features with the input string (query). A ranked list can be generated by scoring matches using a simple *term frequency (TF)* count—the number of matches between the query vector and the n-gram vector of a candidate word. For example, given the string:

"exanple" ⇒ {#exanple#}
⇒ {#ex, exa, xan, anp, npl, ple, le#}

the word "example" is a candidate:

"example" $\Rightarrow$ {#example#}
$\Rightarrow$ {#ex, exa, xam, amp, mpl, ple, le#}

Since the two items share four letter n-grams—"#ex", "exa", "ple", and "le#"—the TF score of the candidate word "example" for the input string "exanple" is four. Note also that the TF score can be used to establish a threshold or cutoff score to limit the number of candidates to consider.

In step 2, the system re-ranks the words in the candidate list using channel probabilities as described above.

On average, the system generates several hundred candidates for a given string. Only the first N candidates are retained for context-based word-error correction.

2.4 The Word Correction System for OCR Post-Processing

The architecture of the word correction system for OCR post-processing is given in Figure 1.

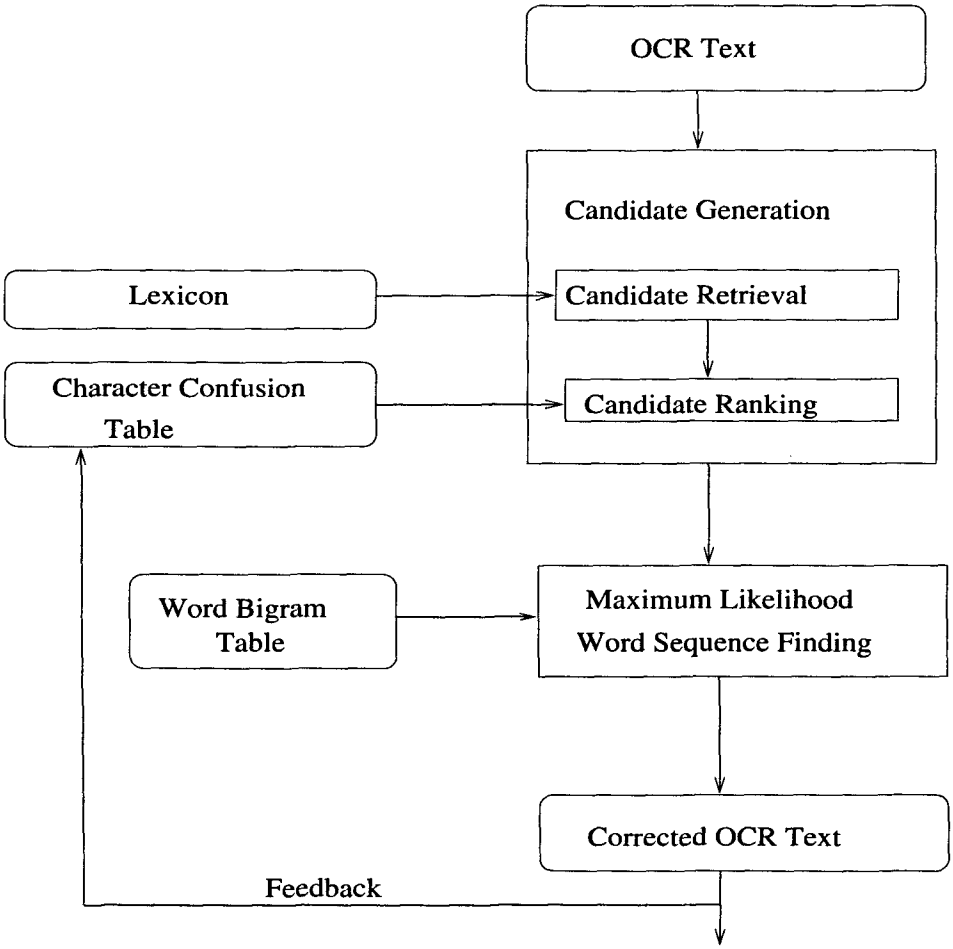


Figure 1: System Architecture

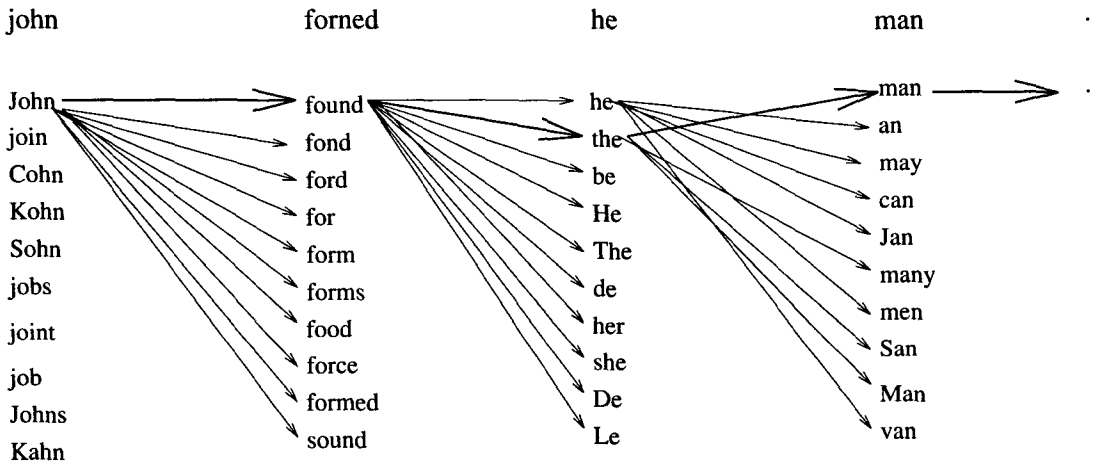
The lexicon is generated from the training text; it includes all the words in the training set with frequency greater than the preset threshold. The words in the lexicon are indexed by letter n-grams as described in the previous section.

The overall process for correcting a sentence is as follows:

1. Read a sentence from the input OCR text.
2. Retrieve up to M candidates from the lexicon for each possible error.[2] Rerank the M candidates by their conditional probabilities to the error. Keep only the top N candidates for the next processing step. (In the current system, M is 10,000 and N is 10.)
3. Use the Viterbi algorithm to get the best word sequence for the strings in the sentence.

Figure 2 illustrates the alternative choices and the optimal path found during the processing (correcting) of the sentence "john fornd he man".

Original Sentence: John found the man.
Input Sentence: john fornd he man.
Corrected Sentence: John found the man.



Best Word Sequence: John found the man .

Figure 2: Process of Correcting a Sentence

The system requires several passes to correct an OCR text. In the first pass, the system has no information on the character confusion probabilities, so it will assume a prior belief α as the probability that a character is correctly recognized. The system distributes the rest of the probability uniformly among other events. (Cf. Formula 9.) In each feedback step, the system first generates a character confusion probability table by comparing the OCR text to the corrected OCR text from the last pass. It uses the new confusion table for the next-pass correction of the OCR text.

[2] In its non-word error mode of operation, the system treats every word that does not match a lexicon entry as a possible error. In its non-word and real-word error mode, the system treats every word as though it were a possible error.

3 Experiments and Results

To test our OCR-error-correction process, we used a set of electronic documents from the Ziff-Davis (ZIFF) news wire.[3] The documents in the corpus are business articles in the domain of computer science and computer engineering. We used 90% of the collection for training and the remaining 10% for testing.

The system created a lexicon and collected word-bigram sequences and statistics from the training data. Words or word-bigrams with frequency less than three were discarded. The resulting lexicon contained about 100,000 words; these were indexed using 34,847 letter n-grams. The resulting word-bigram table had about 1,000,000 entries.

Seventy pages of ZIFF data in the test set were printed in 7-point *Times* font. We degraded the print quality of the documents by photocopying them on a 'light' setting. The photocopies were then scanned by a Fujitsu 3097E scanner and the resulting images were processed by Xerox Textbridge OCR software.

The set of documents contained 55,699 strings and the overall word error rate after OCR processing was 22.9% (12,760). For literal words in the source (only letter sequences, not alpha-numeric ones), the error rate was lower, 14.7% (8,198). Table 1 gives the number of real-word and non-word errors for literal words in the OCR data.

	Non-Word Errors	Real-Word Errors	Total Errors
Number	6,506	1,692	8,198
%	79.4	20.6	100

Table 1: OCR Errors Originating from Literal Words

We conducted three experiments:

1. *Isolated-Word Error Correction*: The system used only channel probabilities without considering context information, i.e., it always selected the candidate with the highest rank in the candidates list to correct a given OCR string.
2. *Context-Dependent Non-Word Error Correction*: The system used context to correct strings that did match valid lexicon words.
3. *Context-Dependent Non- and Real-Word Error Correction*: The system treated all input strings as possible errors and tried to correct them by taking into account the contexts in which the strings appeared.

In each experiment, the system conducted four correction passes: one initial pass with prior probability $\alpha = 0.99$ and three feedback passes.

Results are given in Tables 2, 3, and 4. In all cases, we considered only those strings whose correct forms are literal words (not alpha-numerics). Note that errors can be introduced by the system when it incorrectly changes a correct word in the OCR text into another word. In fact, we distinguish two types of errors introduced by the system: errors caused by changing correct

[3]The ZIFF collection is distributed as part of the data used in the Text Retrieval Conference (TREC) evaluations. The corpus contains about 33 million words.

unknown words and errors caused by changing correct lexicon words. The error reduction rate was calculated by subtracting total errors from 8,198 and dividing by 8,198.

The system, running unoptimized code on a 128MHz DECalpha processor, processed the test corpus at a rate of about 200 words (strings) per second for experiments 1 and 2; and 30 words (strings) per second for experiment 3.

Pass	Non-Word Errors		Real-Word Errors		Introduced Errors		Total Errors	Error Reduction (%)
	Remain	Corrected	Remain	Corrected	Unknown Wds	Lex Wds		
First	3,049	3,457	1,692	0	182	0	4,923	39.9
Feedback-1	2,816	3,690	1,692	0	182	0	4,690	42.8
Feedback-2	2,791	3,715	1,692	0	182	0	4,665	43.1
Feedback-3	2,784	3,722	1,692	0	182	0	4,658	43.2

Table 2: Results from Isolated-Word Error Correction

Pass	Non-Word Errors		Real-Word Errors		Introduced Errors		Total Errors	Error Reduction (%)
	Remain	Corrected	Remain	Corrected	Unknown Wds	Lex Wds		
First	2,684	3,822	1,692	0	182	0	4,558	44.4
Feedback-1	1,972	4,354	1,692	0	182	0	3,846	53.1
Feedback-2	1,943	4,563	1,692	0	182	0	3,817	53.4
Feedback-3	1,948	4,558	1,692	0	182	0	3,822	53.4

Table 3: Results from Context-Dependent Non-Word Error Correction

Pass	Non-Word Errors		Real-Word Errors		Introduced Errors		Total Errors	Error Reduction (%)
	Remain	Corrected	Remain	Corrected	Unknown Wds	Lex Wds		
First	2,529	3,977	1,225	467	182	54	3,990	51.3
Feedback-1	1,978	4,528	1,031	661	182	119	3,310	59.6
Feedback-2	1,935	4,571	1,008	684	182	141	3,266	60.2
Feedback-3	1,926	4,580	1,015	677	182	147	3,270	60.1

Table 4: Results from Context-Dependent Real- and Non-Word Error Correction

4 Analysis

Based on the results, we can see that the predominant, positive effect in correction occurs in the first pass. Performance also improves significantly in the first feedback process, as the system learns the character confusion probabilities by correcting the OCR text. The second and third feedback steps have only slight effect on the error reduction rates. Indeed, in experiment 3, the result from the third feedback pass is actually worse than that from the second feedback pass. These results indicate that an initial pass followed by two feedback passes may optimize the method. In the following discussion, we compare the three experiments using the results obtained from the second feedback step (Feedback-2).

As we might expect, the results from the context-based experiments are much better than those from the isolated-word experiment. The error reduction rates in experiments 2 and 3 are,

respectively, 10.3% and 17.1% higher than the rate in experiment 1. This indicates that even a modest (e.g., bigram-based) representation of context is useful in selecting the best candidates for word-error correction.

In all three experiments, the system introduced 182 new errors due to false corrections of words that were not in the lexicon. (Recall that the system lexicon is based on the words derived from the training corpus; some words may be present in the test corpus that are not in the training corpus.) Whenever the system encounters an unknown word, it treats it as a non-word error and attempts to correct it. In such cases, the system replaces the presumed non-word error with a word from its lexicon. Thus, for example, if the system encounters the word "MobileData" (a correct name) in the OCR output, but does not have "MobileData" in its lexicon, it might change "MobileData" to "MobileComm" (a word that does exist in the training corpus lexicon). Of course, such problems in processing unknown words are not unique to OCR error correction; they represent a general problem for all natural-language processing tasks.

As shown by experiment 3, when the system uses context-based non- and real-word error correction, it achieves a total error reduction rate of 60.2%. This is 6.8% higher than the rate achieved in the context-based non-word experiment. The improvement in performance is gained principally from the reduction of the real-word errors. Although the system introduces additional errors—since all the strings in the OCR text are treated as possible errors and subject to change—the number of corrected real-word errors far exceeds the number of real-word errors introduced. In the second feedback pass, for example, the system introduced 141 new errors by changing correct lexicon words into other lexicon words. On the other hand, the system properly corrected 684 real errors—32.1% of all the real errors. The corrected OCR text, therefore, has 543 fewer real-word errors than the original OCR text.

Certain types of errors in the source or OCR-output text present systematic problems for our approach, highlighting the limitations of the system. In particular, because the process is based on the structural definition of a word (viz., a character sequence 'between white space')—not a morphological one—any errors that obscure word boundaries will defy correction. For example, run-on errors (e.g., "of the"/"ofthe") and split-word errors ("training"/"train ng") cannot be corrected. In addition, the use of a vector-space querying to find candidate lexical entries—including our special approach to word decomposition and scoring—can present problems when processing some OCR errors, especially short strings. For example, if "both" (in the source) is rendered as "hotn" (in the OCR text), it is not possible for the system to generate "both" as one of the high-ranked candidates—they share only one feature, the bigram "ot"— despite the fact that the conditional probability $pr(\text{"hotn"}|\text{"both"})$ might be high. Finally, the system suffers from the common limitation of word bigram or trigram models in that it cannot capture discourse properties of context, such as topic and tense, which are sometimes required to select the correct word.

5 Conclusion

The system we have created uses information from a variety of sources—letter n-grams, character confusion probabilities, and word-bigram probabilities—to realize context-based, automatic, word-error correction. It can correct non-word errors as well as real-word errors. The system can also learn character confusion probability tables by correcting OCR text and use such information to achieve better performance. Overall, for complete (real- and non-word) error correction, it achieved a 60.2% rate of error reduction.

The techniques we have used are subject to certain systematic problems. However, we believe they will prove to be useful not only in improving the quality of OCR processing, but also in enhancing a variety of information retrieval applications.

In future work, we plan to explore different heuristics to deal with word boundary problems and to incorporate other models of context representation, including both SLM approaches, such as word trigram models, and simple discourse structures.

Acknowledgements

We thank Nataša Milić-Frayling and an anonymous reviewer for their excellent comments on an earlier version of this paper. Naturally, the authors alone are responsible for any errors or omissions in the current version.

References

[Atwell & Elliittm 1987] Atwell, E., and Elliittm S. 1987. Dealing with ill-formed English text (Chapter 10). In Garaside, R., Leach, G., and Sampson, G. (eds), *The Computational Analysis of English: A Corpus-Based Approach*. New York: Longman, Inc.

[Charniak 1993] Charniak, E. 1993. *Statistical Language Learning*. MIT Press.

[Church & Gale 1991] Church, K.W., and Gale, W.A. 1991. Probability scoring for spelling correction. *Stat. Comput., 1*, 93–103.

[Dagan & Pereira 1994] Dagan, I., and Pereira, F. 1994. Similarity-based estimation of word cooccurrence probabilities. *Proceedings of the 32nd Annual Meeting of the ACL*, New Mexico State University.

[Golding 1995] Golding, R.A. 1995. A Bayesian hybrid method for context-sensitive spelling correction. *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA. 39–53.

[Golding & Schabes 1996] Golding, R.A., and Schabes, Y. 1996. Combining trigram-based and feature-based methods for context-sensitive spelling correction. *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, CA. (To appear)

[Jelinek 1988] Jelinek, F. 1988. Self-organized language modeling for speech recognition. In Waibel, A., and Lee, K.-F. (eds), *Readings in Speech Recognition*. Morgan Kaufmann Publishers. 450–506.

[Kukich 1992] Kukich, K. 1992. Techniques for automatically correcting words in text. *Comput. Surv.*, 24, 4, 377–439.

[Liu *et al.* 1991] Liu, L.M., Babad, Y.M., Sun, W., Chan, K.K. 1991. Adaptive post-processing of OCR text via knowledge acquisition. *1991 ACM Computer Science Conference. Preparing for the 21st Century*. 558–569.

[Mays *et al.* 1991] Mays, E., Damerau, F.J., and Mercer, R.L. 1991. Context based spelling correction. *Information Processing and Management*, 27, 5, 517–522.

[Salton 1989] Salton, G. 1989. *Automatic Text Processing*. Addison-Wesley Publishing Company.

[Wagner 1974] Wagner, R.A. 1974. The string-to-string correction problem. *J. ACM, 21*, 1, Jan. 1974, 168–173.

A Example of OCR Correction

Original Text

Power-supply IC controls both PWM and power-factor correction.

Designers are focusing more on power-factor correction when creating integrated circuits, due to limited energy supplies, new standards and the type of office electrical loads found in offices. Micro Linear Corp's ML4819 makes the designer's job easier by including both power-faction-correction and PWM control on one chip. This integrated circuit aids in increasing a supply's power factor with fewer components than other implementations. The ML4819 is available in 20-pin DIPs for $3.95 for 100 units. Applications for the product include power supplies for microcomputers in the 150 to 400W range, computer peripherals, instruments, plotters, printers and other off-line power supplies.

OCR Text:

tN˜wer-supp(y IC conimls both PWM and power-factor correciiiifl.

t)esigners are focusing more on power-factor correction when creating integrated circuits. due to limited energy supplies. new ˜andards and the type of office electncal loacs found in iffices. Micro Linear Corp's ML4819 makes the designer's job easier by including both power-faction.correction and PWM control in one chip. This integrated circuit aids in increasing a supply's power factor with Thwer components than other implementations. The ML4819 is available in 20-pm DIPs for 53.95 for ((1) units. Applications for the product include power supplies for microcomputers in the 150 to 41)0W range. computer penpoerals. instirirnents. plotters. pnnters and other off-line power supplies.

Corrected OCR Text from First-Pass Correction:
Note: the correction for a given string is in brackets.

tN˜wer[Newer] - supp(y[supply] IC conimls[coils] both PWM and power - factor correciiiifl[correction] .

t)esigners[Designers] are focusing more on power - factor correction when creating integrated circuits . due to limited energy supplies . new ˜andards[standards] and the type of office electncal[electrical] loacs[loads] found in iffices[offices] . Micro Linear Corp ' s ML4819[XL19] makes the designer ' s job easier by including both power - faction[action] . correction and PWM control in one chip . This integrated circuit aids in increasing a supply ' s power factor with Thwer[fewer] components than other implementations . The ML4819[XL19] is available in 20 - pm[ppm] DIPs for 53.95 for ((1) units . Applications for the product include power supplies for microcomputers in the 150 to 41)0W[NEW] range . computer penpoerals[peripherals] . instirirnents[instruments] . plotters . pnnters[punters] and other off - line power supplies .

Corrected OCR Text from Feedback Correction:
Note: the correction for a given string is in brackets.

tN˜wer[Power] - supp(y[supply] IC conimls[controls] both PWM and power - factor correciiiifl[correction] .

t)esigners[Designers] are focusing more on power - factor correction when creating integrated circuits . due to limited energy supplies . new ˜andards[Standards] and the type of office electncal[electrical] loacs[loads] found in iffices[offices] . Micro Linear Corp ' s ML4819 makes the designer ' s job easier by including both power - faction[factor] . correction and PWM control in one chip . This integrated circuit aids in increasing a supply ' s power factor with Thwer[Tower] components than other implementations .The ML4819 is available in 20 - pm[pin] DIPs for 53.95 for ((1) units . Applications for the product include power supplies for microcomputers in the 150 to 41)0W[ROW] range . computer penpoerals[peripherals] . instirirnents[instruments] . plotters . pnnters[printers] and other off - line power supplies .