A Generative Probabilistic OCR Model

Okan Kolak[†,⋆] **Philip Resnik**[‡,⋆]
Computer Sci.[†], Linguistics[‡], UMIACS[⋆]
University of Maryland
College Park, MD 20742, USA
{okan,resnik}@umiacs.umd.edu

William Byrne
CLSP
The Johns Hopkins University
Baltimore, MD 21218, USA
byrne@jhu.edu

Abstract

In this paper, we present a generative probabilistic optical character recognition (OCR) model that describes an end-to-end process from generation of the true word sequence to the noisy output of an OCR system. The model is designed for use in error correction in a post-processing framework, treating the OCR system as a black-box. The focus of the system is to make OCR output more useful for computer usage. We present a £nite-state machine based implementation of the model, and demonstrate its ability to signi£cantly reduce word and character error rates.

1 Introduction

Despite the increase in the amount of text stored in electronic form, vast quantities of information is still available primarily, or only, in print. Some important applications of the NLP technology, such as rapid, rough document translation in the £eld [1] or information retrieval from scanned documents [2], can depend heavily on the quality of optical character recognition (OCR) output. The alternative of using the raw images remains to be elusive for practical systems [3].

Unfortunately, the output of commercial OCR systems is far from perfect, especially when the language in question is resource-poor [4]. And efforts to acquire new language resources from hardcopy using OCR [5] face something of a chicken-and-egg problem. The problem is compounded by the fact that most OCR system are black boxes that do not allow user tuning or re-training

In this paper, we describe a complete, probabilistic, generative model for OCR, motivated speci£cally by (a) the need to deal with monolithic OCR systems, (b) the focus on OCR as a component in NLP applications, and (c) the ultimate goal of using OCR to help acquire resources for new languages from printed text. After presenting the model itself, we discuss the model's implementation, training, and its use for post-OCR error correction. We provide evalution results for error correction; and conclude with a discussion of related research and directions for future work. Kolak et al. [6] provides a more complete description and evaluation of the work presented here.

2 The Model

We propose a generative model that relate an observable OCR output string O to an underlying true word sequence W. Probability of this relationship, $P(W, O)$, is decomposed by Bayes's Rule into steps modeled by $P(W)$ (the source model) and $P(O|W)$ (comprising sub-steps generating O from W). Each step and substep is completely modular, so one can ¤exibly make use of existing sub-models or devise new ones as necessary. Note that the process of "generating" O from W is a mathematical abstraction, not necessarily related to the operation of any particular OCR system.

We begin with preliminary de£nitions and notation, illustrated in Figure 1. A true word sequence $W =$

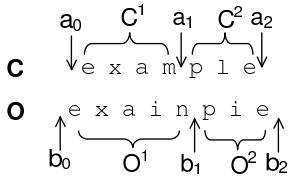

Figure 1: Word and character segmentation

$\langle W_1, \ldots, W_r \rangle$ corresponds to a true character sequence $C = \langle C_1, \ldots, C_n \rangle$, and the OCR system's output character sequence is given by $O = \langle O_1, \ldots, O_m \rangle$.

A segmentation of the true character sequence into p subsequences is represented as $\langle C^1, \ldots, C^p \rangle$. Segment boundaries are only allowed between characters. Subsequences are denoted using segmentation positions $a = \langle a_1, \ldots, a_{p-1} \rangle$, where $a_i < a_{i+1}$, $a_0 = 0$, and $a_p = n$. The a_i de£ne character subsequences $C^i = (C_{a_{i-1}}, \ldots, C_{a_i}]$. (The number of segments p need not equal the number of words r and C^i need not be a word in W.)

Correspondingly, a segmentation of the OCR'd character sequence into q subsequences is given by $\langle O^1, \ldots, O^q \rangle$. Subsequences are denoted by $b =$

$\langle b_1, \ldots, b_{q-1} \rangle$, where $b_j < b_{j+1}$, $b_0 = 0$, and $b_q = m$. The b de£ne character subsequences $O^j = (O_{b_{j-1}} \ldots O_{b_j}]$.

Alignment chunks are pairs of corresponding truth and OCR subsequences: $\langle O^i, C^i \rangle$, $i = 1, \ldots, p$.

Generation of True Word Sequence. The generative process begins with production of the true word sequence W with probability $P(W)$; for example, $W = \langle this, is, an, example, . \rangle$. Modeling the underlying sequence at the word level facilitates integration with NLP models, which is our ultimate goal. For example, the distribution $P(W)$ can be de£ned using n-grams, parse structure, or any other language modeling tool.

From Words to Characters. The £rst step in transforming W to O is generation of a character sequence C, modeled as $P(C|W)$. This step accommodates the character-based nature of OCR systems. It also allows mapping of different character sequences to the same word sequence (case/font variation) or vice versa (e.g. ambiguous word segmentation in Chinese). We output '#' to represent visible word boundaries (i.e. spaces). One possible C for our example W is C = "This#is#an#example."

Segmentation. Subsequences C^i are generated from C by choosing a set of boundary positions, a. This substep, modeled by $P(a|C, W)$, is motivated by the fact that most OCR systems £rst perform image segmentation, and then perform recognition on a word by word basis. Word merge and split errors are modeled at this step as well. A possible segmentation for our example is $a = \langle 8, 11, 13 \rangle$, i.e. C^1 = "This#is#", C^2 = "#an#", C^3 = "#ex", C^4 = "ample." Notice the merge error in segment 1 and the split error involving segments 3 and 4.

Character Sequence Transformation. Our characterization of the £nal step, transformation into an observed character sequence, is motivated by the need to model OCR systems' character-level recognition errors. We model each subsequence C^i as being transformed into an OCR subsequence O^i, so

$$P(O, b|a, C, W) = P(\langle O^1, \ldots, O^q \rangle | a, C, W). \quad (1)$$

and we assume each C^i is transformed independently, allowing

$$P(\langle O^1, \ldots, O^q \rangle | a, C, W) \approx \prod_{i=1}^{p} P(O^i|C^i). \quad (2)$$

Any character-level string error model can be used to de£ne $P(O^i|C^i)$; this is also a logical place to make use of con£dence values if provided by the OCR system. For our example C^i, a possible result for this step is: O^1 = "Tlmsis", O^2 = "an", O^3 = "cx", O^4 = "amp1e."; $b = \langle 7, 10, 13 \rangle$. The £nal generated string would therefore be O = "Tlmsis#an#cx#am1e.".

Assuming independence of the individual steps, the complete model estimates joint probability

$$P(O, b, a, C, W) = P(O, b|a, C, W)P(a|C, W)P(C|W)P(W) \quad (3)$$

$P(O, W)$ can be computed by summing over all possible b, a, C that can transform W to O:

$$P(O, W) = \sum_{b,a,C} P(O, b, a, C, W). \quad (4)$$

3 Implementation

We have implemented the generative model using a weighted £nite state model (FSM) framework, which provides a strong theoretical foundation, ease of integration for different components, and reduced implementation time thanks to available toolkits such as the AT&T FSM Toolkit [7]. Each step is represented and trained as a separate FSM, and the resulting FSMs are then composed together to create a single FSM that encodes the whole model. Details of parameter estimation and decoding follow.

3.1 Parameter Estimation

The parameter estimation methods we devised assume that a training corpus is available, containing $\langle O, C, W \rangle$ triples. Speci£c parameter estimation methods for each individual step is desribed in the following sections.

Generation of True Word Sequence. We use an n-gram language model generated using CMU-Cambridge Toolkit.[8] as the source model for the original word sequence. The model is trained on the W from the training data and encoded as a simple FSM. We made a closed vocabulary assumption to evaluate the effectivenes of our model when all correct words are in its lexicon. Therefore, although the language model is trained on only the training data, the words in the test set are included in the £nal language model FSM.

From Words to Characters. We generate three different character sequence variants for each word: upper case, lower case, and leading case (e.g. `this` ⇒ {`THIS`, `this`, `This`}). For each word, the distribution over case variations is learned from the $\langle W, C \rangle$ pairs in the training corpus. For words with very low or zero occurence counts, we back off to word-independent case variant probabilities.

Segmentation. Our current implementation makes an independent decision for each character pair whether to insert a boundary between them. The number of boundary insertions are limited to one per word for practical reasons. The probability of inserting a segment boundary between two characters is conditioned on the character pair, and estimated from the training corpus.

Table 1: Post-correction WER and CER and their reduction rates under various conditions

Conditions				Results			
LM	WC	SG	EM	WER (%)	Red. (%)	CER (%)	Red. (%)
Original OCR Output				18.31	-	5.01	-
Unigram	3 options	None	Sect. 9	7.41	59.53	3.42	31.74
Unigram	3 options	None	Sect. 1-9	7.12	61.11	3.35	33.13
Unigram	3 options	None	Sect. 5-9	7.11	61.17	3.34	33.33
Trigram	3 options	None	Sect. 5-9	7.06	61.44	3.32	33.73
Trigram	Best case	2 way	Sect. 5-9	6.75	63.13	2.91	41.92

Character Sequence Transformation. This step is implemented as a probabilistic string edit process. The confusion tables for edit operations are estimated using Viterbi style training on $\langle O, C \rangle$ pairs in training data. Our current implementation allows for substitution, deletion, and insertion errors, and does not use context characters.

Final Cleanup. At this stage, special symbols that were inserted into the character sequence are removed and the £nal output sequence is formed. For instance, segment boundary symbols are removed or replaced with spaces depending on the language.

3.2 Decoding

Decoding is the process of £nding the "best" W for an observed $(\hat{O}, \hat{b})$, namely

$$\hat{W} = \underset{W}{\operatorname{argmax}}\{\max_{a,C}[P(\hat{O}, \hat{b}|a, C, W)P(a|C, W)P(C|W)P(W)]\} \quad (5)$$

Decoding within the FSM framework is straightforward: we simply compose all the components of the model in order, and then invert the resulting FSM. This produces a single transducer that takes a sequence of OCR characters as input, and returns all possible sequences of truth words as output, along with their weights. The most probable sequence among those returned by the composition can be used as the output of the post-OCR correction process. Alternatively, the resulting lattice or N-best list can easily be integrated with other probabilistic models over words.

4 Experimental Evaluation

Although most researchers are interested in improving the results of OCR on degraded documents, we are primarily interested in developing and improving OCR in new languages for use in NLP. A possible approach to retargeting OCR for a new language is to employ an existing OCR system from a "nearby" language, and then to apply our error correction framework. For these experiments, therefore, we created our experimental data by scanning a hardcopy Bible using both an English and a French OCR system. (See Kanungo et al. [4] and Resnik et al. [9] for discussion of the Bible as a resource for multilingual OCR and NLP.) We have used the output of the English system run on French input to simulate the situation where available resources of one language are used to acquire resources in another language that is similar.

It was necessary to pre-process the data in order to eliminate the differences between the on-line version that we used as the ground truth and the hardcopy, such as footnotes, glossary, cross-references, page numbers, etc. We have not corrected hyphenations, case differences, etc.

Our evaluation metrics for OCR performance are Word Error Rate (WER) and Character Error Rate (CER), which are de£ned as follows. The results reported are obtained by ignoring character case.

$$\text{WER}(W_{truth}, W_{OCR}) = \frac{WordEditDistance(W_{truth}, W_{OCR})}{|W_{truth}|} \quad (6)$$

$$\text{CER}(C, O) = \frac{CharEditDistance(C, O)}{|C|} \quad (7)$$

Since we are interested in recovering the original word sequence rather than the character sequence, evaluations are performed on lowercased and tokenized data. Note, however, our system works on the original case OCR data, and generates a sequence of word ids, that are converted to a lowercase character sequence for evaluation.

We have divided the data, which has 29317 lines, into 10 equal size disjoint sets, and used the £rst 9 as the training data, and the £rst 500 lines of the last one as the test data. The WER and CER for the English OCR system on the French the test data were 18.31% and 5.01% respectively. The numbers drop to 17.21% and 4.28% when single character tokens and tokens with no alphabetical characters are ignored. The WER and CER for the output generated on French by the French OCR system were 5.98% and 2.11%.

4.1 Reduction of OCR Error Rates

We evaluated the performance of our model by studying the reduction in WER and CER after correction. The input to the system was original case, tokenized OCR output, and the output of the system was a sequence of word ids that are converted to lowercase character sequences for evaluation.

Table 2: WER, CER, and reduction rates ignoring single characters and non-alphabetical tokens

Conditions				Results			
LM	WC	SG	EM	WER (%)	Red. (%)	CER (%)	Red. (%)
Original OCR Output				17.21	-	4.28	-
Unigram	3 options	None	Sect. 9	3.97	76.93	1.68	60.75
Unigram	3 options	None	Sect. 1-9	3.62	78.97	1.60	62.62
Unigram	3 options	None	Sect. 5-9	3.61	79.02	1.58	63.08
Trigram	3 options	None	Sect. 5-9	3.52	79.55	1.56	63.55
Trigram	Best case	2 way	Sect. 5-9	3.15	81.70	1.14	73.36

All the results are summarized in Table 1. The conditions side gives various parameters for each experiment. The language model (LM) is either (word) unigram or trigram. Word to character conversion (WC) can allow 3 case variations that are mentioned before, or simply pick the most probable one for each word. Segmentation (SG) can be disabled, or 2-way split and merges may be allowed. Finally, the character level error model (EM) may be trained on various subsets of training data. Table 2 gives the adjusted results when ignoring all single characters and tokens that do not contain any alphabetical character.

As can be seen from the tables, as we increase the training size of character error model from 1 section to 5 sections, the performance increases. However there is a slight decrease in performance when the training size is increased to 9 sections. This suggests that our training procedures, while effective, may require re£nement as additional training data becomes available. When we replace the unigram language model with a trigram one, the results improve as expected. However, the most interesting case is the last experiment where 2-way word merge/split errors are allowed.

Word merge/split errors cause an exponential increase in the search space; when there are n words that needs to be corrected together, there are N^n possible combinations where N is the vocabulary size. We imposed various restrictions on the model to reduce the search space and achieve acceptable execution times. Despite the imposed restrictions, the ability to handle word merge/split errors improves performance signi£cantly.

5 Related Work

There has been considerable research on automatically correcting words in text in general, and correction of OCR output in particular. Kukich [10] provides a general survey of the research in the area. Unfortunately, there is no commonly used evaluation base for OCR error correction, making comparison of experimental results dif£cult.

Some systems integrate the post-processor with the actual character recognizer to allow interaction between the two. In an early study, Hanson et al. [11] reports a word error rate of about 2% and a reject rate of 1%, without a dictionary. Sinha and Prasada [12] achieves 97% word recognition, ignoring punctuation, using an augmented dictionary, a Viterbi style algorithm, and manual heuristics.

Many systems treat OCR as a black box, generally employing word and/or character level n-grams along with character confusion probabilities. Srihari et al. [13] is one typical example and reports up to 87% error correction on arti£cial data, and relying (as we do) on a lexicon for correction. Goshtasby and Ehrich [14] presents a method that based on probabilistic relaxation labeling, using context characters to constrain the probability of each character. They do not use a lexicon but do require the probabilities assigned to individual characters by the OCR system.

Jones et al. [15] describe an OCR post-processing system comparable to ours, and reports error reductions of 70-90%. Their system is designed around a strati£ed algorithm. The £rst phase performs isolated word correction using rewrite rules, allowing words that are not in the lexicon. The second phase attempts correcting word split errors, and the last phase uses word bigram probabilities to improve correction. In comparison to our work, the main difference is our focus on an end-to-end generative model versus their strati£ed algorithm centered around correction.

Pal et al. [16] describes a method for OCR error correction of an in¤ectional Indian language using morphological parsing, and reports correcting 84% of the words with a single character error. Although it is limited to single errors, the systems demonstrates the possibility of correcting OCR errors in morphologically rich languages.

Although segmentation errors have been addressed to some degree in previous work, to the best of our knowledge our model is the £rst that explicitly incorporates segmentation. Similarly, many systems make use of a language model, a character confusion model, etc., but none have developed an end-to-end model that formally describes the OCR process from the generation of the true word sequence to the output of the OCR system in a manner that allows for statistical parameter estimation. Our model is also the £rst to explicitly model the conversion of a sequence of words into a character sequence.

6 Conclusions and Future Work

We have presented a ¤exible, modular, probabilistic generative OCR model designed speci£cally for ease of integration with probabilistic models of the sort commonly found in recent NLP work, and for rapid retargeting of OCR and NLP technology to new languages.

In a rigorous evaluation of post-OCR error correction on real data, illustrating a scenario where a black-box commercial English OCR system is retargeted to work with French data, we obtained a 70% reduction in word error rate over the English-on-French baseline, with a resulting word accuracy of 97%. It is worth noting that our post-OCR correction of the English OCR on French text led to better performance than a commercial French OCR system run on the same text.

We are currently working on improving the correction performance of the system, and extending our error model implementation to include character context and allow for character merge/split errors. We also intend to relax the requirement of having a word list, so that the model handles valid word errors.

Finally, we plan to challenge our model with other languages, starting with Arabic, Turkish, and Chinese. Arabic and Turkish have phonetic alphabets, but also pose the problem of rich morphology. Chinese will require more work due to the size of its alphabet. We are optimistic that the power and ¤exibility of our modeling framework will allow us to develop the necessary techniques for these languages, as well as many others.

Acknowledgements

This research was supported in part by National Science Foundation grant EIA0130422, Department of Defense contract RD-02-5700, DARPA/ITO Cooperative Agreement N660010028910, and Mitre agreement 010418-7712.

We are grateful to Mohri et al. for the AT&T FSM Toolkit, Clarkson and Rosenfeld for CMU-Cambridge Toolkit, and David Doermann for providing the OCR output and useful discussion.

References

[1] Jim Krane. New handheld translators could be used in Iraq war. *The Honolulu Advertiser*, 2002. October 7.

[2] W. B. Croft, S. M. Harding, K. Taghva, and J. Borsack. An evaluation of information retrieval accuracy with simulated OCR output. In *Symposium of Document Analysis and Information Retrieval, ISRI-UNLV*, 1994.

[3] David Doermann. The indexing and retrieval of document images: A survey. *Computer Vision and Image Understanding: CVIU*, 70(3):287–298, 1998.

[4] Tapas Kanungo, Philip Resnik, Song Mao, Doe wan Kim, and Qigong Zheng. The bible, truth, and multilingual optical character recognition. in revision.

[5] David Doermann, Huanfeng Ma, Burcu Karagöl-Ayan, and Douglas W.Oard. Translation lexicon acquisition from bilingual dictionaries. In *Ninth SPIE Symposium on Document Recognition and Retrieval*, San Jose, CA, 2002.

[6] Okan Kolak, William Byrne, and Philip Resnik. A generative probabilistic OCR model for NLP applications. In *Human Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Alberta, Canada, May 2003. To appear.

[7] Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. A rational design for a weighted £nite-state transducer library. *Lecture Notes in Computer Science*, 1436, 1998.

[8] Philip Clarkson and Ronald Rosenfeld. Statistical language modeling using the CMU-Cambridge Toolkit. In *ESCA Eurospeech*, 1997.

[9] Philip Resnik, Mari Broman Olsen, and Mona Diab. The Bible as a parallel corpus: Annotating the 'Book of 2000 Tongues'. *Computers and the Humanities*, 33:129–153, 1999.

[10] Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24 (4):377–439, December 1992.

[11] Allen R. Hanson, Edward M. Riseman, and Edward G. Fisher. Context in word recognition. *Pattern Recognition*, 8:33–45, 1976.

[12] R. M. K. Sinha and Biendra Prasada. Visual text recognition through contextual processing. *Pattern Recognition*, 21(5):463–479, 1988.

[13] Sargur N. Srihari, Jonathan J. Hull, and Ramesh Choudhari. Integrating diverse knowledge sources in text recognition. *ACM Transactions on Of£ce Information Systems*, 1(1):68–87, January 1983.

[14] Ardeshir Goshtasby and Roger W. Ehrich. Contextual word recognition using probabilistic relaxation labeling. *Pattern Recognition*, 21(5):455–462, 1988.

[15] Mark A. Jones, Guy A. Story, and Bruce W. Ballard. Integrating multiple knowledge sources in a bayesian OCR post-processor. In *IDCAR-91*, pages 925–933, St. Malo, France, 1991.

[16] U. Pal, P. K. Kundu, and B. B. Chaudhuri. OCR error correction of an in¤ectional indian language using morphological parsing. *Journal of Information Science and Engineering*, 16(6):903–922, November 2000.