Proceedings of HLT-NAACL 2003
Main Papers , pp. 55-62
Edmonton, May-June 2003

A Generative Probabilistic OCR Model for NLP Applications

Okan Kolak
Computer Science and UMIACS
University of Maryland
College Park, MD 20742, USA
okan@umiacs.umd.edu

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

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD 20742, USA
resnik@umiacs.umd.edu

Abstract

In this paper, we introduce a generative probabilistic optical character recognition (OCR) model that describes an end-to-end process in the noisy channel framework, progressing from generation of true text through its transformation into the noisy output of an OCR system. The model is designed for use in error correction, with a focus on post-processing the output of black-box OCR systems in order to make it more useful for NLP tasks. We present an implementation of the model based on finite-state models, demonstrate the model's ability to significantly reduce character and word error rate, and provide evaluation results involving automatic extraction of translation lexicons from printed text.

1 Introduction

Although a great deal of text is now available in electronic form, vast quantities of information still exist primarily (or only) in print. Critical applications of NLP technology, such as rapid, rough document translation in the field (Holland and Schlesiger, 1998) or information retrieval from scanned documents (Croft et al., 1994), can depend heavily on the quality of optical character recognition (OCR) output. Doermann (1998) comments, "Although the concept of a raw document image database is attractive, comprehensive solutions which do not require complete and accurate conversion to a machine-readable form continue to be elusive for practical systems."

Unfortunately, the output of commercial OCR systems is far from perfect, especially when the language in question is resource-poor (Kanungo et al., in revision). And efforts to acquire new language resources from hardcopy using OCR (Doermann et al., 2002) 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 — Baird (1999, reported in (Frederking, 1999)) comments that the lack of ability to rapidly retarget OCR/NLP applications to new languages is "largely due to the monolithic structure of current OCR technology, where language-specific constraints are deeply enmeshed with all the other code."

In this paper, we describe a complete probabilistic, generative model for OCR, motivated specifically 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 then present two evaluations: one for standalone OCR correction, and one in which OCR is used to acquire a translation lexicon from printed text. We conclude with a discussion of related research and directions for future work.

2 The Model

Generative "noisy channel" models relate an observable string O to an underlying sequence, in this case recognized character strings and underlying word sequences W. This relationship is modeled by $P(W, O)$, 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 sub-step is completely modular, so one can flexibly make use of existing sub-models or devise new ones as necessary.[1]

We begin with preliminary definitions and notation, illustrated in Figure 1. A true word sequence $W = \langle W_1, \ldots, W_r \rangle$ corresponds to a true character sequence

[1] Note that the process of "generating" O from W is a mathematical abstraction, not necessarily related to the operation of any particular OCR system.

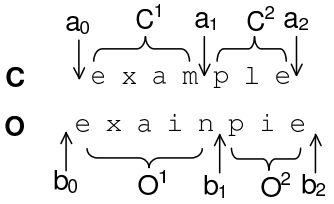

Figure 1: Word and character segmentation

$C = \langle C_1, \ldots, C_n \rangle$, 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 define 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 define character subsequences $O^j = (O_{b_{j-1}} \ldots O_{b_j}]$.

Alignment chunks are pairs of corresponding truth and OCR subsequences: $\langle O^i, C^i \rangle$, $i = 1, \ldots, p$.[2]

2.1 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 defined using n-grams, parse structure, or any other tool in the language modeling arsenal.

2.2 From Words to Characters

The first 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, and provides a place to model the mapping of different character sequences to the same word sequence (case/font variation) or vice versa (e.g. ambiguous word segmentation in Chinese). If the language in question provides explicit word boundaries (e.g. words are separated by spaces when printed) then we output '#' to represent visible word boundaries. One possible C for our example W is C = "This#is#an#example."

[2]The model is easily modified to permit $p \neq q$.

2.3 Segmentation

Subsequences C^i are generated from C by choosing a set of boundary positions, a. This sub-step, modeled by $P(a|C, W)$, is motivated by the fact that most OCR systems first perform image segmentation, and then perform recognition on a word by word basis.

For a language with clear word boundaries (or reliable tokenization or segmentation algorithms), one could simply use spaces to segment the character sequence in a non-probabilistic way. However, OCR systems may make segmentation errors and resulting subsequences may or may not be words. Therefore, a probabilistic segmentation model that accommodates word merge/split errors is necessary.

If a segment boundary coincides with a word boundary, the word boundary marker '#' is considered a part of the segment on both sides. 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.

2.4 Character Sequence Transformation

Our characterization of the final 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

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

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

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

Any character-level string error model can be used to define $P(O^i|C^i)$; for example Brill and Moore (2000) or Kolak and Resnik (2002). This is also a logical place to make use of confidence values if provided by the OCR system. We assume that # is always deleted (modeling merge errors), and can never be inserted. Boundary markers at segment boundaries are re-inserted when segments are put together to create O, since they will be part of the OCR output (not as #, but most likely as spaces). 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 final generated string would therefore be O = "Tlmsis#an#cx#am1e.".

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

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

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

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

3 Implementation

We have implemented the generative model using a weighted finite 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 (Mohri et al., 1998). 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 specific model definition and estimation methods assume that a training corpus is available, containing $\langle O, C, W \rangle$ triples.

Generation of True Word Sequence. We use an n-gram language model as the source model for the original word sequence: an open vocabulary, trigram language model with back-off generated using CMU-Cambridge Toolkit (Clarkson and Rosenfeld, 1997). The model is trained on the W from the training data using the Witten-Bell discounting option for smoothing, and encoded as a simple FSM. We made a closed vocabulary assumption to evaluate the effectiveness 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 language model FSM, and treated as unseen vocabulary.

From Words to Characters. We generate three different character sequence variants for each word: upper case, lower case, and leading case (e.g. `this` $\Rightarrow$ {`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 that do not appear in the corpus, or do not have enough number of occurrences to allow a reliable estimation, we back off to word-independent case variant probabilities.[3]

Segmentation. Our current implementation makes an independent decision for each character pair whether to insert a boundary between them. To reduce the search space associated with the model, we limit the number of boundary insertions to one per word, allowing at most two-way word-level splits. The probability of inserting a segment boundary between two characters, conditioned on the character pair, is estimated from the training corpus, with Witten-Bell discounting (Witten and Bell, 1991) used to handle unseen character pairs.

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.[4] Figure 2 shows a fragment of a weighted FSM model for $P(O^i|C^i)$: it shows how the observed O^i `haner` could be generated by underlying C^i `banker` or `hacker`.[5]

Final Cleanup. At this stage, special symbols that were inserted into the character sequence are removed and the final 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 finding the "best" W for an observed $(\hat{O}, \hat{b})$, namely

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

Decoding within the FSM framework is straightforward: we first 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. One can then simply encode OCR character sequences as FSMs and compose them with the model transducer to perform decoding.

Note that the same output sequence can be generated through multiple paths, and we need to sum over all paths to find the overall probability of that sequence. This can be achieved by determinizing the output FSM generated by the decoding process. However, for practical reasons, we chose to first find the N-best paths in the resulting FSM and then combine the ones that generate the same output.

The resulting lattice or N-best list is easily integrated with other probabilistic models over words, or the most

[3] Currently, we assume a Latin alphabet. Mixed case text is not included since it increases the number of alternatives drastically; at run time mixed-case words are normalized as a preprocessing step.

[4] We are working on conditioning on neighbor characters, and using character merge/split errors. These extensions are trivial conceptually, however practical constraints such as the FSM sizes make the problem more challenging.

[5] The probabilities are constructed for illustration, but realistic: notice how `n` is much more likely to be confused for `c` than `k` is.

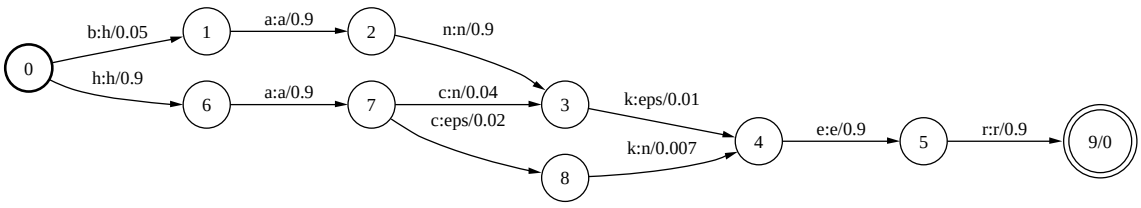

Figure 2: Fragment of an FSM for $P(O^i|C^i)$.

probable sequence can be used as the output of the post-OCR correction process.

4 Experimental Evaluation

We report on two experiments. In the first, we evaluate the correction performance of our model on real OCR data. In the second, we evaluate the effect of correction in a representative NLP scenario, acquiring a translation lexicon from hardcopy text.

4.1 Training and Test Data

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. (in revision) and Resnik et al. (1999) 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. 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 defined as follows:

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

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

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, that 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 first 9 as the training data, and the first 500 lines of the last one as the test data.[6] The WER and CER for the English OCR system on the French test data were 18.31% and 5.01% respectively. The error rates were 5.98% and 2.11% for the output generated by the French OCR system on the same input. When single characters and non-alphabetical tokens are ignored, the WER and CER drop to 17.21% and 4.28% for the English OCR system; 4.96% and 1.68% for the French OCR system.

4.2 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.

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 the three case variations mentioned earlier, or simply pick the most probable variant for each word. Segmentation (SG) can be disabled, or 2-way splits and merges may be allowed. Finally, the character level error model (EM) may be trained on various subsets of training data.[7] 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 the character error model from one section to five sections, the performance increases. However, there

[6] Each line contains a verse, so they can actually span several lines on a page.

[7] Other sub-models are always trained on all 9 training sections.

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

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				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

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

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 refinement as additional training data becomes available. When we replace the unigram language model with a trigram model, the results improve as expected. However, the most interesting case is the last experiment, where word merge/split errors are allowed.

Word merge/split errors cause an exponential increase in the search space. If there are n words that needs to be corrected together, they can be grouped in 2^{n-1} different ways; ranging from n distinct tokens to a single token. For each of those groups, there are $N^{m \cdot k}$ possible correct word sequences where m is the number of tokens in that group, k is the maximum number of words that can merge together, and N is the vocabulary size. Although it is possible to avoid some computation using dynamic programming, doing so would require some deviation from the FSM framework.

We have instead used several restrictions to reduce the search space. First, we allowed only 2-way merge and split errors, restricting the search space to bigrams. We further reduce the search space by searching through only the bigrams that are seen in the training data. We also introduced character error thresholds, letting us eliminate candidates based on their length. For instance, if we are trying to correct a sequence of 10 characters and have set a threshold of 0.2, we only need check candidates whose length is between 8 and 12. The last restriction we imposed is to force selection of the most likely case for each word rather than allowing all three case variations. Despite all these limitations, the ability to handle word merge/split errors improves performance significantly.

It is notable that our model allows global interactions between the distinct components. As an example, if the input is "ter- re", the system returns "mer se" as the most probable correction. When "la ter- re" is given as the input, interaction between the language model, segmentation model, and the character error model chooses the correct sequence "la terre". In this example, the language model overcomes the preference of the segmentation model to insert word boundaries at whitespaces.

4.3 Translation Lexicon Generation

We used the problem of unsupervised creation of translation lexicons from automatically generated word alignment of parallel text as a representative NLP task to evaluate the impact of OCR correction on usability of OCR text. We assume that the English side of the parallel text is online and its foreign language translation is generated using an OCR system.[8] Our goal is to apply our OCR error correcting procedures prior to alignment so the resulting translation lexicon has the same quality as if it had been derived from error-free text.

We trained an IBM style translation model (Brown et al., 1990) using GIZA++ (Och and Ney, 2000) on the 500 test lines used in our experiments paired with corresponding English lines from an online Bible. Word level alignments generated by GIZA++ were used to extract cross-language word co-occurrence frequencies, and candidate

[8] Alternatively, the English side can be obtained via OCR and corrected.

translation lexicon entries were scored according to the log likelihood ratio (Dunning, 1993) (cf. (Resnik and Melamed, 1997)).

We generated three such lexicons by pairing the English with the French ground truth, uncorrected OCR output, and its corrected version. All text was tokenized, lowercased, and single character tokens and tokens with no letters were removed. This method of generating a translation lexicon works well; as Table 3 illustrates with the top twenty entries from the lexicon generated using ground truth French.

and	et		for	car
of	de		if	si
god	dieu		ye	vous
we	nous		you	vous
christ	christ		the	le
not	pas		law	loi
but	mais		jesus	jésus
lord	seigneur		as	comme
the	la		that	qui
is	est		in	dans

Table 3: Translation lexicon entries extracted using ground truth French

Figure 3 gives the precision-recall curves for the translation lexicons generated from OCR using the English OCR system on French hardcopy input with and without correction, using the top 1000 entries of the lexicon generated from ground truth as the target set. Since we are interested in the effect of OCR, independent of the performance of the lexicon generation method, the lexicon auto-generated from the ground truth provides a reasonable target set. (More detailed evaluation of translation lexicon acquisition is a topic for future work.)

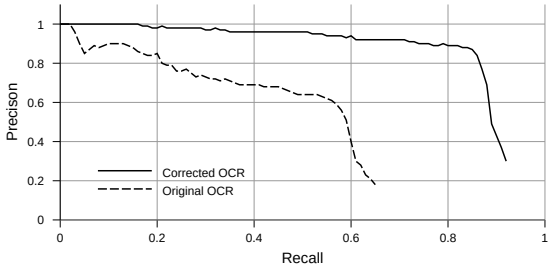

Figure 3: Effect of correction on translation lexicon acquisition

The graph clearly illustrates that the precision of the translation lexicon generated using original OCR data degrades quickly as recall increases, whereas the corrected version maintains its precision above 90% up to a recall of 80%.

5 Related Work

There has been considerable research on automatically correcting words in text in general, and correction of OCR output in particular. Kukich (1992) 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 difficult.

Some systems integrate the post-processor with the actual character recognizer to allow interaction between the two. In an early study, Hanson et al. (1976) reports a word error rate of about 2% and a reject rate of 1%, without a dictionary. Sinha and Prasada (1988) achieve 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. (1983) is one typical example and reports up to 87% error correction on artificial data, relying (as we do) on a lexicon for correction. Goshtasby and Ehrich (1988) presents a method 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. (1991) describe an OCR post-processing system comparable to ours, and report error reductions of 70-90%. Their system is designed around a stratified algorithm. The first 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. The three phases interact with each other to guide the search. In comparison to our work, the main difference is our focus on an end-to-end generative model versus their stratified algorithm centered around correction.

Perez-Cortes et al. (2000) describes a system that uses a stochastic FSM that accepts the smallest k-testable language consistent with a representative language sample. Depending on the value of k, correction can be restricted to sample language, or variations may be allowed. They report reducing error rate from 33% to below 2% on OCR output of hand-written Spanish names from forms.

Pal et al. (2000) describes a method for OCR error correction of an inflectional 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 system demonstrates the possibility of correcting OCR errors in morphologically rich languages.

Taghva and Stofsky (2001) takes a different approach

to post-processing and proposes an interactive spelling correction system specifically designed for OCR error correction. The system uses multiple information resources to propose correction candidates and lets the user review the candidates and make corrections.

Although segmentation errors have been addressed to some degree in previous work, to the best of our knowledge our model is the first 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 first to explicitly model the conversion of a sequence of words into a character sequence.

6 Conclusions and Future Work

We have presented a flexible, modular, probabilistic generative OCR model designed specifically 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 also evaluated the impact of error correction in a resource-acquisition scenario involving translation lexicon acquisition from OCR output. The results show that our post-OCR correction framework significantly improves performance. We anticipate applying the technique in order to retarget cross-language IR technology — the results of Resnik et al. (2001) demonstrate that even noisy extensions to dictionary-based translation lexicons, acquired from parallel text, can have a positive impact on cross language information retrieval performance.

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.

We are also exploring the possibility of tuning a statistical machine translation model to be used with our model to exploit parallel text. If a translation of the OCR'd text is available, a translation model can be used to provide us with a candidate-word list that contains most of the correct words, and very few irrelevant words.

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 character set. We are optimistic that the power and flexibility of our modeling framework will allow us to develop the necessary techniques for these languages, as well as many others.

Acknowledgments

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

Eric Brill and Robert C. Moore. 2000. An improved model for noisy channel spelling correction. In *38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong, China, October.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

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

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

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

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

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, March.

Robert Frederking. 1999. Summary of the MIDAS session on handling multilingual speech, document images, and video OCR, August. http://www.clis2.umd.edu/conferences/midas/papers/frederking.txt.

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

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

Melissa Holland and Chris Schlesiger. 1998. High-modality machine translation for a battlefield environment. In *NATO/RTO Systems Concepts and Integration Symposium*, Monterey, CA, April. 15/1-3. Hull, Canada: CCG, Inc. (ISBN 92-837-1006-1).

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

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

Okan Kolak and Philip Resnik. 2002. OCR error correction using a noisy channel model. In *Human Language Technology Conference (HLT 2002)*, San Diego, CA, March.

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

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

Franz. J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *ACL00*, pages 440–447, Hongkong, China, October.

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

Juan Carlos Perez-Cortes, Juan-Carlos Amengual, Joaquim Arlandis, and Rafael Llobet. 2000. Stochastic error-correcting parsing for OCR post-processing. In *ICPR*, pages 4405–4408, Barcelona, Spain, September.

Philip Resnik and I. Dan Melamed. 1997. Semi-automatic acquisition of domain-specific translation lexicons. In *Fifth Conference on Applied Natural Language Processing*, Washington, D.C.

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

Philip Resnik, Douglas Oard, and Gina Levow. 2001. Improved cross-language retrieval using backoff translation. In *Human Language Technology Conference (HLT-2001)*, San Diego, CA, March.

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

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

Kazem Taghva and Eric Stofsky. 2001. OCRSpell: an interactive spelling correction system for OCR errors in text. *IJDAR*, 3(3):125–137.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1093, July.