OCR Post-Processing for Low Density Languages

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

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

Abstract

We present a lexicon-free post-processing method for optical character recognition (OCR), implemented using weighted finite state machines. We evaluate the technique in a number of scenarios relevant for natural language processing, including creation of new OCR capabilities for low density languages, improvement of OCR performance for a native commercial system, acquisition of knowledge from a foreign-language dictionary, creation of a parallel text, and machine translation from OCR output.

1 Introduction

The importance of rapidly retargeting existing natural language processing (NLP) technologies to new languages is widely accepted (Oard, 2003). Statistical NLP models have a distinct advantage over rule based approaches to achieve this goal, as they require far less manual labor; however, training statistical NLP methods requires on-line text, which can be hard to find for so-called "low density" languages — that is, languages where few on-line resources exist. In addition, for many languages of interest input data are available mostly in printed form, and must be converted to electronic form prior to processing.

Optical character recognition (OCR) is often the only feasible method to perform this conversion, owing to its speed and cost-effectiveness. Unfortunately, the performance of OCR systems is far from perfect and recognition errors significantly degrade the performance of NLP applications. This is true both in resource acquisition, such as automated bilingual lexicon generation (Kolak et al., 2003), and for end-user applications such as rapid machine translation (MT) in the battlefield for document filtering (Voss and Ess-Dykema, 2000). Moreover, for low density languages, there simply may not be an OCR system available.

In this paper, we demonstrate that via statistical post-processing of existing systems, it is possible to achieve reasonable recognition accuracy for low density languages altogether lacking an OCR system, to significantly improve on the performance of a trainable commercial OCR system, and even to improve significantly on a native commercial OCR system.[1] By taking a post-processing approach, we require minimal assumptions about the OCR system used as a starting point.

The proper role of our post-processing approach depends on the language. For languages with little commercial potential for OCR, it may well provide the most practical path for language-specific OCR development, given the expensive and time consuming nature of OCR development for new languages and the "black box" nature of virtually all state-of-the-art OCR systems. For languages where native OCR development may take place, it is a fast, practical method that allows entry into a new language until native OCR development catches up. For these, and also for languages where native systems exist,

[1]Currently we assume the availability of an OCR system that supports the script of the language-of-interest, or which is script independent (Natarajan et al., 2001).

we show that post-processing can yield improvements in performance.

Sections 2 and 3 describe the method and its implementation. In Section 4 we cover a variety of relevant NLP scenarios: Creating OCR capabilities for Igbo, performing OCR on a dictionary for Cebuano, using OCR to acquire the Arabic side of a common parallel text, and evaluating the value of OCR post-processing for machine translation of Arabic and Spanish. In Sections 5 and 6 we discuss related work and summarize our findings.

2 Post-Processing System

We use the noisy channel framework to formulate the correction problem, revising our previous model (Kolak et al., 2003). That model takes the form

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

whose components are a word-level source model $P(W)$, a word-to-character model $P(C|W)$, a segmentation model $P(a|C, W)$, and a model for character sequence transformation, $P(O, b|a, C, W)$. W is the correct word sequence and C is the corresponding character sequence, which is recognized as O by the OCR system. a and b are segmentation vectors for C and O.

The original model requires a lexicon that covers all words in the processed text — a strong assumption, especially for low density languages. We converted the model into a character-based one, removing the need for a lexicon. Generation of W is replaced by generation of C, which renders $P(C|W)$ irrelevant, and the model becomes

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

Although word-based models generally perform better, moving from words to characters is a necessary compromise because word-based models are useless in the absence of a lexicon, which is the case for many low-density languages.

In addition to eliminating the need for a lexicon, we developed a novel method for handling word merge/split errors.[2] Rather than modeling these errors explicitly using a segmentation model, we simply treat them as character deletion/insertion errors involving the space character, allowing us to handle them within the error model. The segmentation step is absorbed into the character transformation step, so a and b are no longer necessary, hence the final equation becomes

$$P(O, C) = P(O|C)P(C)$$

which is a direct application of the noisy channel model. We can describe the new generative process as follows: First, a sequence of characters C are generated, with probability $P(C)$, and the OCR system converts it into O with probability $P(O|C)$. For example, if the actual input was $a\llcorner car$ and it was recognized as $ajar$, $P(ajar, a\llcorner car) = P(ajar|a\llcorner car)P(a\llcorner car)$. Using the channel model to address word merge/split errors without actually using a word level model is, to our knowledge, a novel contribution of our approach.

3 Implementation

We implemented our post-processing system using the framework of weighted finite state machines (WFSM), which provides a strong theoretical foundation and reduces implementation time, thanks to freely available toolkits, such as the AT&T FSM Toolkit (Mohri et al., 1998). It also allows easy integration of our post-processor with numerous NLP applications that are implemented using FSMs (e.g. (Knight and Graehl, 1997; Kumar and Byrne, 2003)).

3.1 Source Model

The source model assigns probability $P(C)$ to original character sequences, C. We use character level n-gram language models as the source model, since n-gram models are simple, easy to train, and usually achieve good performance. More complicated models that make use of constraints imposed by a particular language, such as vowel harmony, can be utilized if desired. We used the CMU-Cambridge Language Modeling Toolkit v2 (Clarkson and Rosenfeld, 1997) for training, using Witten-Bell smoothing and vocabulary type 1; all other parameters were left at their default values.

[2] A merge error occurs when two or more adjacent items are recognized as one, and a split error occurs when an item is recognized as two or more items. These errors can happen both at word level and character level.

3.2 Channel Model

The channel model assigns a probability to O given that it was generated from C. We experimented with two probabilistic string edit distance models for implementing the channel model. The first, following our earlier model (2003), permits single-character substitutions, insertions, and deletions, with associated probabilities. For example, $P(ajar|a\text{␣}car) \approx P(a \mapsto a)P(\text{␣} \mapsto \epsilon)P(c \mapsto j)P(a \mapsto a)P(r \mapsto r)$. Note that we are only considering the most likely edit sequence here, as opposed to summing over all possible ways to convert $a\text{␣}car$ to $ajar$. The second is a slightly modified version of the spelling correction model of Brill and Moore (2000).[3] This model allows many-to-many edit operations, which makes $P(liter|litre) \approx P(l \mapsto l)P(i \mapsto i)P(tre \mapsto ter)$ possible. We will refer to the these as the single-character (SC) and multi-character (MC) error models, respectively.

We train both error models over a set of corresponding ground truth and OCR sequences, $\langle C, O \rangle$. Training is performed using expectation-maximization: We first find the most likely edit sequence for each training pair to update the edit counts, and then use the updated counts to re-estimate edit probabilities. For MC, after finding the most likely edit sequence, extended versions of each non-copy operation that include neighboring characters are also considered, which allows learning any common multi-character mappings. Following Brill and Moore, MC training performs only one iteration of expectation-maximization.

In order to reduce the time and space requirements of the search at correction time, we impose a limit on number of errors per token. Note that this is not a parameter of the model, but a limit required by its computational complexity. A lower limit will almost always result in lower correction performance, so the highest possible limit allowed by time and memory constraints should be used. It is possible to correct more errors per token by iterating the correction process. However, iterative correction cannot guarantee that the result is optimal under the model.

[3]We ignore the location of the error within the word, since it is not as important for OCR as it is for spelling.

3.3 Chunking

Since we do not require a lexicon, we work on lines of text rather than words. Unfortunately the search space for correcting a complete line is prohibitively large and we need a way to break it down to smaller, independent chunks. The chunking step is not part of the model, but rather a pre-processing step: chunks are identified, each chunk is corrected independently using the model, and the corrected chunks are put back together to generate the output.

Spaces provide a natural break point for chunks. However, split errors complicate the process: if parts of a split word are placed in different chunks, the error cannot be corrected. For example, in Figure 1, chunking (b) allows the model to produce the desired output, but chunking (a) simply does not allow combining "sam" and "ple" into "sample", as each chunk is corrected independently.

give	a sam	ple case

(a)

give a	sam ple	case

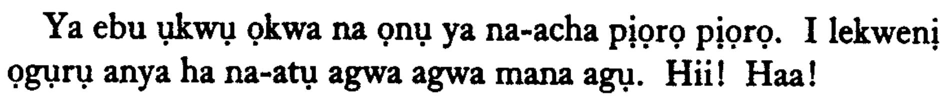
(b)

Figure 1: Example of a bad and a good chunking

We address this by using the probabilities assigned to spaces by the source model for chunking. We break the line into two chunks using the space with the highest probability and repeat the process recursively until all chunks are reduced to a reasonable size, as defined by time and memory limitations. Crucially, spurious spaces that cause split errors are expected to have a low probability, and therefore breaking the line using high probability spaces reduces the likelihood of placing parts of a split word in different chunks.

If a lexicon does happen to be available, we can use it to achieve more reliable chunking, as follows. The tokens of the input line that are present in the lexicon are assumed to be correct. We identify runs of out-of-lexicon tokens and attempt to correct them together, allowing us to handle split errors. Note that in this case the lexicon is used only to improve chunking, not for correction. Consequently, coverage of the lexicon is far less important.

Our lexicon-free chunking algorithm placed an erroneous boundary at 11.3% of word split points for Arabic test data (Section 4.3). However, correction performance was identical to that of error-



Ya ebu ụkwụ ọkwa na ọnụ ya na-acha pịọrọ pịọrọ. I lekwenị ọgụrụ anya ha na-atụ agwa agwa mana agụ. Hii! Haa!

Figure 2: A small excerpt from Akụkọ

free chunking.[4] Incorrect decisions did not hurt because the correction method was not able to fix those particular split errors, regardless. The errors of the chunking and correction models coincided as they both rely on the same language model. Therefore, chunking errors are unlikely to reduce the correction performance.

3.4 Correction

Correction is performed by estimating the most probable source character sequence $\hat{C}$ for a given observed character sequence O, using the formula:

$$\hat{C} = \underset{C}{\operatorname{argmax}}\{P(O|C)P(C)\}$$

We first encode O as an FSA and compose it with the inverted error model FST.[5] The resulting FST is then composed with the language model FSA. The final result is a lattice that encodes all feasible sequences C, along with their probabilities, that could have generated O. We take the sequence associated with the most likely path through the lattice as $\hat{C}$.

4 Evaluation

We evaluate our work on OCR post-processing in a number of scenarios relevant for NLP, including creation of new OCR capabilities for low density languages, improvement of OCR performance for a native commercial system, acquisition of knowledge from a foreign-language dictionary, creation of a parallel text, and machine translation from OCR output. The languages studied include Igbo, Cebuano, Arabic, and Spanish.

For intrinsic evaluation, we use the conventional Word Error Rate (WER) metric, which is defined as

$$WER(C,O) = \frac{WordEditDistance(C,O)}{WordCount(C)}$$

[4]Ignoring errors that result in valid words, lexicon-based chunking is always error-free.

[5]Inversion reverses the direction of the error model, mapping observed sequences to possible ground truth sequences.

We do not use the Character Error Rate (CER) metric, since for almost all NLP applications the unit of information is the words. For extrinsic evaluation of machine translation, we use the BLEU metric (Papineni et al., 2002).

4.1 Igbo: Creating an OCR System

Igbo is an African language spoken mainly in Nigeria by an estimated 10 to 18 million people, written in Latin script. Although some Igbo texts use diacritics to mark tones, they are not part of the official orthography and they are absent in most printed materials. Other than grammar books, texts for Igbo, even hardcopy, are extremely difficult to obtain. To our knowledge, the work reported here creates the first OCR system for this language.

For the Igbo experiments, we used two sources. The first is a small excerpt containing 6727 words from the novel "Juo Obinna" (Ubesie, 1993). The second is a small collection of short stories named "Akụkọ Ife Nke Ndị Igbo" (Green and Onwuamaegbu, 1970) containing 3544 words. We will refer to the former as "Juo" and the latter as "Akụkọ" hereafter. We generated the OCR data using a commercial English OCR system.[6] Juo image files were generated by scanning 600dpi laser printer output at 300dpi resolution. Akụkọ image files were generated by scanning photocopies from the bound hardcopy at 300dpi. Figure 2 provides a small excerpt from the actual Akụkọ page images used for recognition. For both texts, we used the first two thirds for training and the remaining third for testing.

We trained error and language models (EMs and LMs) using the training sets for Juo and Akụkọ separately, and performed corrections of English OCR output using different combinations of these models on both test sets. Table 1 shows the results for the Juo test set while Table 2 presents the results for Akụkọ. The relative error reduction ranges from 30% to almost 80%. The SC error model performs better than the MC error model under all conditions.

[6]Abby Fine Reader Professional Edition Version 7.0

Conditions			Results	
LM Data	EM Data	EM type	WER (%)	Red. (%)
Juo	Juo	MC	8.66	74.18
Juo	Akụkọ	MC	15.23	54.59
Akụkọ	Juo	MC	13.25	60.49
Akụkọ	Akụkọ	MC	19.08	43.11
Juo	**Juo**	**SC**	**7.11**	**78.80**
Juo	Akụkọ	SC	11.49	65.74
Akụkọ	Juo	SC	13.42	59.99
Akụkọ	Akụkọ	SC	18.92	43.59
Original OCR Output			**35.44**	-

Table 1: Post-correction WER for English OCR on Juo

Conditions			Results	
LM Data	EM Data	EM type	WER (%)	Red. (%)
Juo	Juo	MC	21.42	36.33
Juo	Akụkọ	MC	18.08	46.25
Akụkọ	Juo	MC	21.51	36.06
Akụkọ	Akụkọ	MC	18.16	46.02
Juo	Juo	SC	19.92	40.78
Juo	Akụkọ	SC	16.49	50.98
Akụkọ	Juo	SC	19.92	40.78
Akụkọ	**Akụkọ**	**SC**	**16.40**	**51.25**
Original OCR Output			**33.64**	-

Table 2: Post-correction WER for English OCR on Akụkọ

This is due to the fact that MC requires more training data than SC. Furthermore, most of the errors in the data did not require many-to-many operations. Results in Tables 1 and 2 are for 6-gram language model and error limit of 5; corresponding 3-gram error rates were 1% to 2% (absolute) higher.

The best correction performance is achieved when both the EM and LM training data come from the same source as the test data, almost doubling the performance achieved when they were from a different source.[7] Note that the amount of training data is small, four to eight pages, so optimizing performance via manual entry of document-specific training text is not unrealistic for scenarios involving long documents such as books.

4.1.1 Using a Trainable OCR System

In an additional experiment with Igbo, we found that post-processing can improve performance substantially even when an OCR system trained on Igbo characters is the starting point. In particular, the commercial OCR system used for Igbo experiments supports user-trained character shape models. Using Juo as the source, we trained the commercial OCR system manually on Igbo characters, resulting in a 7.43% WER on Juo without postprocessing.[8] Note that this is slightly higher than the 7.11% WER achieved using an English OCR system together with our post-processing model. We used a 6-gram LM, and a SC EM with error limit of 5. Table 3 shows that by post-processing the Igbo-trained OCR system, we reduce the word error rate by 50%.

Conditions		Results	
LM Data	EM Data	WER (%)	Red. (%)
Juo	**Juo**	**3.69**	**50.34**
Juo	Akụkọ	5.24	29.48
Akụkọ	Juo	5.08	31.63
Akụkọ	Akụkọ	7.38	0.67
Original OCR Output		**7.43**	-

Table 3: Post-correction WER for trained OCR system on Juo

4.2 Cebuano: Acquiring a Dictionary

Cebuano is a language spoken by about 15 million people in the Philippines, written in Latin script. The scenario for this experiment is converting a Cebuano hardcopy dictionary into electronic form, as in DARPA's Surprise Language Dry Run (Oard, 2003). The dictionary that we used had diacritics, probably to aid in pronunciation. The starting-point OCR data was generated using a commercial OCR system.[9] The fact that the tokens to be corrected come from a dictionary means (1) there is little context available and (2) word usage frequencies are not reflected. Character-based models may be affected by these considerations, but probably not to the extent that word-based models would be.

Table 4 shows WER for Cebuano after post-processing. The *size* column represents the number of dictionary entries used for training, where each entry consists of one or more Cebuano words. As can be seen from the table, our model reduces WER substantially for all cases, ranging from 20% to 50% relative reduction. As expected, the correction performance increases with the amount of training data; note, however, that we achieve reasonable correction performance even using only 500 dictionary entries for training.

[7]There was no overlap between training and test data under any circumstance.

[8]The system trains by attempting OCR on a document and asking for the correct character whenever it is not confident.

[9]ScanSoft Developer's Kit 2000, which has no built-in support for Cebuano.

Conditions			Results	
Size	LM	EM	WER (%)	Red. (%)
500	3-gram	SC	5.37	33.04
500	3-gram	MC	5.05	37.03
500	6-gram	SC	6.41	20.07
500	6-gram	MC	5.33	33.54
1000	3-gram	SC	5.33	33.54
1000	3-gram	MC	4.63	42.27
1000	6-gram	SC	5.58	30.42
1000	6-gram	MC	4.67	41.77
27363	3-gram	SC	4.34	45.89
27363	3-gram	MC	4.14	48.38
27363	6-gram	SC	4.55	43.27
27363	**6-gram**	**MC**	**3.97**	**50.50**
Original OCR Output			**8.02**	-

Table 4: Post-correction WER for Cebuano

Contrary to the Igbo results, the MC error model performs better than the SC error model. And, interestingly, the 3-gram language model performs better than the 6-gram model, except for the largest training data and MC error model combination. Both differences are most likely caused by the implications of using a dictionary as discussed above.

4.3 Arabic: Acquiring Parallel Text

We used Arabic to illustrate conversion from hardcopy to electronic text for a widely available parallel text, the Bible (Resnik et al., 1999; Kanungo et al., 2005; Oard, 2003). We divided the Bible into ten equal size segments, using the first segment for training the error model, the first nine segments for the language model, and the first 500 verses from the last segment for testing. Since diacritics are only used in religious text, we removed all diacritics. The OCR data was generated using a commercial Arabic OCR system.[10] Note that this evaluation differs from Igbo and Cebuano, as the experiments were performed using an existing *native* OCR system. It also allowed us to evaluate chunking, as Arabic data has far more word merge/split errors compared to Igbo and Cebuano.

Table 5 shows the correction performance for Arabic under various conditions. The *Limit* column lists the maximum number of errors per token allowed and the *M/S* column indicates whether correction of word merge/split errors was allowed. We achieve significant reductions in WER for Arabic. The first two rows show that the 6-gram language model performs much better than the 3-gram model. Interestingly, higher order n-grams perform worse when we allow word merge/split errors. Note that for handling word merge/split errors we need to learn the character distributions within lines, rather than within words as we normally do. Consequently, more training data is required for reliable parameter estimation. Handling word merge/split errors improve the performance, which is expected. Allowing fewer errors per token reduces the performance, since it is not possible to correct words that have more character errors than the limit. Unfortunately, increasing the error limit increases the search space exponentially, making it impossible to use high limits. As mentioned in Section 3.2, iterative correction is a way to address this problem.

[10]Sakhr Automatic Reader Version 6.0

Conditions			Results	
M/S	LM	Limit	WER (%)	Red. (%)
no	3-gram	2	22.14	10.33
no	6-gram	2	17.99	27.14
yes	3-gram	2	18.26	26.04
yes	**3-gram**	**4**	**17.74**	**28.15**
yes	5-gram	2	20.74	16.00
Original OCR Output			**24.69**	-

Table 5: Post-correction WER for Arabic

4.4 Extrinsic Evaluation: MT

While our post-processing methods reduce WER, our main interest is their impact on NLP applications. We have performed machine translation experiments to measure the effects of OCR errors and the post-processing approach on NLP application performance.

For Arabic, we trained a statistical MT system using the first nine sections of the Bible data. The language model is trained using the CMU-Cambridge toolkit and the translation model using the GIZA++ toolkit (Och and Ney, 2000). We used the ReWrite decoder (Germann, 2003) for translation.

BLEU scores for OCR, corrected, and clean text were 0.0116, 0.0141, and 0.0154, respectively. This establishes that OCR errors degrade the performance of the MT system, and we are able to bring the performance much closer to the level of performance on clean text by using post-processing. Clearly the BLEU scores are quite low; we are planning to perform experiments on Arabic using a more advanced translation system, such as Hiero (Chiang, 2005).

MT System	Input Text	BLEU Score
Systran	OCR	0.2000
Systran	Corrected	0.2606
Systran	Clean	0.3188
ReWrite	OCR	0.1792
ReWrite	Corrected	0.2234
ReWrite	Clean	0.2590

Table 6: Spanish-English translation results

In order to test in a scenario with better translation performance, we performed MT evaluations using Spanish. We used a commercial translation system, Systran, in addition to statistical translation. More resources being available for this language, corrected text for Spanish experiments was obtained using our original model that takes advantage of a lexicon (2003). Table 6 shows that scores are much higher compared to Arabic, but the pattern of improvements using post-processing is the same.

5 Related Work

There has been considerable research on automatic error correction in text. Kukich (1992) provides a general survey of the research in the area. Unfortunately, there is no standard evaluation benchmark for OCR correction, and implementations are usually not publicly available, making a direct comparison difficult.

Most correction methods are not suitable for low density languages as they rely on lexicons. Goshtasby and Ehrich (1988) present a lexicon-free method based on probabilistic relaxation labeling. However, they use the probabilities assigned to individual characters by the OCR system, which is not always available. Perez-Cortes et al. (2000) describe a method which does not have this limitation. They use a stochastic FSM that accepts the smallest k-testable language consistent with a representative sample. While the method can handle words not in its lexicon in theory, it was evaluated using a large k to restrict corrections to the lexicon. They report reducing error rate from 33% to below 2% on OCR output of hand-written Spanish names.

In addition to providing alternatives, the literature provides complementary methods. Guyon and Pereira (1995) present a linguistic post-processor based on variable memory length Markov models that is designed to be used as the language model component of character recognizers. Their model can be used as the source model for our method. Since it is a variable length model, it can allow us to handle higher order n-grams.

A script-independent OCR system is presented by Natarajan et al. (2001). The system is evaluated on Arabic, Chinese, and English, achieving 0.5% to 5% CER under various conditions. Since our post-processing method can be used to reduce the error rate of a trained OCR system, the two methods can be combined to better adapt to new languages.

Voss and Ess-Dykema (2000) evaluated the effects of OCR errors on MT in the context of the FALCon project, which combines off-the-shelf OCR and MT components to provide crude translations for filtering. They report significant degradation in translation performance as a result of OCR errors. For instance, for the Spanish system, OCR process reduced the number of words that can be recognized by the translation module by more than 60%.

6 Conclusions

We have presented a statistical post-processing method for OCR error correction that requires minimal resources, aimed particularly at low density languages and NLP scenarios. The technique gains leverage from existing OCR systems, enabling both minimal-labor adaptation of systems to new low density languages and improvements in native OCR performance.

We rigorously evaluated our approach using real OCR data, and have shown that we can achieve recognition accuracy lower than that achieved by a trainable OCR system for a new language. For Igbo, a very low density language, adapting English OCR achieved relative error reductions as high as 78%, resulting in 7.11% WER. We also showed that the error rate of a trainable OCR system after training can be further reduced up to 50% using post-processing, achieving a WER as low as 3.7%. Post-processing experiments using Cebuano validate our approach in a dictionary-acquisition scenario, with a 50.5% relative reduction in error rate from 8.02% to 3.97%. Evaluation on Arabic demonstrated that the error rate for a native commercial OCR system can be reduced by nearly 30%. In addition, we measured the impact of post-processing on machine translation, quantifying OCR degradation of MT performance and showing that our technique moves the performance of MT on OCR data significantly closer to performance on clean input. See Kolak (forthcoming) for more details and discussion.

One limitation of our approach is its reliance on an existing OCR system that supports the script of the language of interest. Trainable OCR systems are the only option if there is no OCR system that supports the script of interest; however, training an OCR system from scratch is usually a tedious and time consuming task. Post-processing can be used to reduce the training time and improve recognition accuracy by aiding generation of more training data once basic recognition capability is in place.

Acknowledgments

This research was supported in part by Department of Defense contract RD-02-5700 and DARPA/ITO Cooperative Agreement N660010028910. We are grateful to Mohri et al. for the AT&T FSM Toolkit and Clarkson and Rosenfeld for the CMU-Cambridge Toolkit. We thank David Doermann and his students for the Cebuano text; Chinedu Uchechukwu for the Igbo text and useful information on the language. We also thank Mustafa Murat Tıkır for his help in generating Igbo ground truth for Akụkọ, and for useful discussion.

References

Eric Brill and Robert C. Moore. 2000. An improved model for noisy channel spelling correction. In *Proceedings of the ACL-00*, Hong Kong, China, October.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL-05*, pages 263–270, Ann Arbor, Michigan, USA, June.

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

Ulrich Germann. 2003. Greedy decoding for statistical machine translation in almost linear time. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

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

M. M. Green and M. O. Onwuamaegbu, editors. 1970. *Akụkọ Ife Nke Ndị Igbo*. Oxford University Press, Ibadan, Nigeria.

Isabelle Guyon and Fernando Pereira. 1995. Design of a linguistic postprocessor using variable memory length Markov models. In *Proceedings of the ICDAR-95*, volume 1, Montreal, Quebec, Canada, August.

Tapas Kanungo, Philip Resnik, Song Mao, Doe wan Kim, and Qigong Zheng. 2005. The Bible and multilingual optical character recognition. *Communications of the ACM*, 48(6):124–130.

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the ACL-97*, Madrid, Spain, July.

Okan Kolak, William Byrne, and Philip Resnik. 2003. A generative probabilistic OCR model for NLP applications. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

Okan Kolak. forthcoming. *Cross-Lingual Utilization of NLP Resources for New Languages*. Ph.D. thesis, University of Maryland, College Park, Maryland, USA.

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

Shankar Kumar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the HLT-NAACL-03*, Edmonton, Alberta, Canada, May.

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.

Premkumar Natarajan, Zhidong Lu, Richard Schwartz, Issam Bazzi, and John Makhoul. 2001. Multilingual machine printed ocr. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):43–63.

Douglas W. Oard. 2003. The surprise language exercises. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(2):79–84, June.

Franz. J. Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the ACL-00*, pages 440–447, Hongkong, China, October.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the ACL-02*, Philedelphia, Pennsylvania, USA, July.

Juan Carlos Perez-Cortes, Juan-Carlos Amengual, Joaquim Arlandis, and Rafael Llobet. 2000. Stochastic error-correcting parsing for OCR post-processing. In *Proceedings of the ICPR-00*, Barcelona, Spain, September.

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(1-2):129–153.

Tony Uchenna Ubesie. 1993. *Juo Obinna*. University Press PLC, Ibadan, Nigeria. ISBN: 19575395X.

Clare R. Voss and Carol Van Ess-Dykema. 2000. When is an embedded MT system 'good enough' for filtering? In *Proceedings of the Workshop on Embedded MT Systems, ANLP-NAACL-00*, Seattle, Washington, USA, May.