Improving Book OCR by Adaptive Language and Image Models

Dar-Shyang Lee
Google Inc.
dsl@google.com

Ray Smith
Google Inc.
rays@google.com

Abstract—**In order to cope with the vast diversity of book content and typefaces, it is important for OCR systems to leverage the strong consistency within a book but adapt to variations across books. In this work, we describe a system that combines two parallel correction paths using document-specific image and language models. Each model adapts to shapes and vocabularies within a book to identify inconsistencies as correction hypotheses, but relies on the other for effective cross-validation. Using the open source Tesseract engine as baseline, results on a large dataset of scanned books demonstrate that word error rates can be reduced by 25% using this approach.**

Keywords: document-specific OCR; adaptive OCR; error correction

I. Introduction

The coming of digital libraries has inspired research interests in building book adaptive OCR systems that would work across a large diversity of content and typefaces. In addition to training on an ever increasing amount of data, it has been generally accepted that the strong redundancy and consistency within a large scan collection could be leveraged to improve system accuracy and robustness.

Book adaptive OCR shares much similarity with speaker adaptation in speech recognition[5]. With a suitable choice of classifier architecture, parameters of a base book-independent model can be adjusted over unlabeled test samples under maximum likelihood or MAP criteria using EM [7,10]. However, the approach is not generally applicable to arbitrary classifiers.

Modern OCR engines often adopt a learning-on-the-fly approach by selecting a set of reliable decisions produced by the static classifier for retraining [8]. Adaptation can also be done as a postprocess by selecting reliable words to retrain a special classifier [2]. This approach depends critically on the success of selecting the correct words for adaptation.

Some research goes further by starting without any static shape classifier at all and seeks to automatically decode the text by exploring shape clusters and iteratively propagating label assignment driven by a language model[1,3]. These deciphering solutions are ideal for dealing with rare fonts that are atypical of any training samples, but are insufficient for achieving high precision performance on their own.

The above methods can be characterized as image-based correction driven by a global language model. The accuracy of these image-based adaptive methods hinges on the adequacy of the underlying language model. Limited research has simultaneously adapted *both* the image and language models on a book. Xiu & Baird[11] minimize the global mutual-entropy between an image-based iconic model and a word-based linguistic model to optimize whole book recognition. Unfortunately, such a global optimization strategy converges slowly.

In this work, we combine both the image-based adaptive approach and language-based correction to independently detect inconsistencies in the model for correction, but rely on cross-validation by the other model for verification. This is motivated by the fact that many similarly shaped confusions such as m/rn, and 1/l/I are difficult to reliably separate across all fonts, but often easily distinguishable from context. On the other hand, language-based correction would be ineffective in choosing between "black" and "blue", which are easily distinguishable from the image. The goal is to utilize the strength of each model to resolve the most obvious confusions. In addition, we allow both the image model and language model to adapt to the document to be more robust to content and typeface diversity. We demonstrate that the document-specific language and image correction working in tandem improves significantly over the baseline open source OCR engine Tesseract[8].

II. System Overview

A. Design Motivation

Our system design was motivated by several key factors. First, the observation that most confusions are ambiguous only in image or in language suggests that a cross model verification would resolve the most obvious cases. Instead of optimizing a global cost function over the consistency and complexity of both image and language models simultaneously, we took a simpler but more efficient approach to let each correction path operate independently, coupled only loosely by cross-validation.

Secondly, while most research focuses on shape adaptation to cope with variations in font styles and scanning characteristics driven by a global static language model, we noticed that document vocabularies differ significantly and topic keywords are often unfairly demoted due to a low prior probability in the base distribution. In order to handle the large diversity of book contents and typefaces, both the image model and language model must be adaptive.

As is the case with all adaptive systems, the question is how to decide which answers are reliable enough to adapt on and how to use that information to improve the model. The overall control strategy and system components are described next.

B. System Control

Our system consists of two analogous correctional paths, one image-based and one language-based, shown in Fig.1. In the former, similar-shaped fragments are considered to be the same class. Therefore, shapes that look more similar to clusters with a different label than to clusters with the same label are suspicious. The conflicts can be resolved by consulting other similarly-shaped clusters. The shape clusters and labels should also be consistent when expanded to larger context to include their neighboring symbols.

A similar approach could be applied to language-based correction. Word tokens which are spelled the same, independent of surrounding punctuation marks and surface forms, are considered the same class. This token, when evaluated across all occurrences in contexts, should have the highest likelihood. Out-of-vocabulary tokens are considered suspicious and evaluated across all instances to find if a more likely answer exists. This could generate valid spelling corrections but also false hypotheses in ambiguous linguistic context. Each instance of correction is evaluated by the image model to reject those obvious mismatches.

After verification, the accepted changes are updated in the output file and document models. The language model gets a new list of verified words, and the image model needs to update the shape clusters. The specific corrections made are also accumulated to keep track of common confusions. It is obvious that this process is iterative by nature and there could be many variants of the exact sequence. We have found that the exact sequence of operations (eg. synchronous update of all changes on both models) does not make a big difference. In addition, although we were able to get additional improvements through more iterations, most of the performance gain was realized in the first iteration.

C. System Components

The image model is composed of a static (base) shape model and a dynamic (adaptive) model. The base shape model is usually trained on labeled data for a large number of fonts and serves as the baseline for all books, whereas the adaptive model relies on unsupervised learning to construct document-specific templates. For the purpose of our discussion, the base model is the OCR engine, which could be invoked to make general shape classification. The adaptive model is constructed by clustering character shapes grouped by labels assigned by the base model. By exploring the cluster structure and comparing characters with their neighbors, we can identify inconsistencies between labels and shapes. As mentioned, there has been much research on performing this type of adaptation to correct the labels, but our shape model has the unique feature of utilizing shapes of bigrams as well as shapes of individual characters, which helps to correct segmentation errors made by the original OCR engine. We rely on the language model to verify if a correction is warranted.

The language model is analogous to the image model in many ways. It also consists of a static base model and an adaptive document (cache) model. The base model is trained on all the data for the language (or sub-collection), and the cache model is constructed from the somewhat dubious labels produced by OCR. The reason for using a cache model is that vocabularies vary greatly from book to book. A single base model often unfairly penalizes against topic-specific words that appear with high frequency in a book. In our case, the base model is a large word n-gram model with back-off to a character n-gram model for out-of-vocabulary (OOV) words. The cache model is constructed from high confidence words and words that have been verified by the base language model and image model. For each questionable word, we evaluate all its occurrences in context jointly to identify potential corrections. These hypotheses are then verified by the image model to determine if they are accepted.

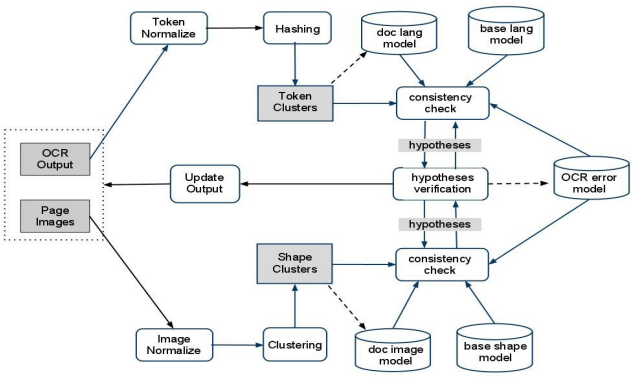

Figure 1. Proposed system of two parallel correction paths based on document-specific image and language models.

III. IMAGE-BASED CORRECTION

Construction of the Adaptive Image Model has two phases: shape clustering and shape classification.

A. Shape Clustering

First, individual component shapes are extracted using class labels and bounding boxes generated by the OCR engine. Each word image identified by the OCR engine is processed to enhance the greyscale image by auto-inversion to make the text black, and contrast enhancement to use the full 8-bit range. Individual and adjacent pairs of characters (the *clips*) are cut out of the cleaned word image, using the bounding boxes from OCR, for clustering.

Each class label and size (to within a couple of pixels) is handled separately by Shape Clustering. A feature vector is constructed with the pixel locations in the clip as dimensions. Since the clips are of slightly different sizes, all pixel locations are computed relative to the centroid of the clip. The clips are clustered using a kd-tree-based hierarchical agglomerative clustering (KDHAC). The KDHAC pivots a different dimension of the input feature vector at each level in the tree, but in this application, the number of dimensions far exceeds the number of levels that will be needed in the kd-tree, so the dimensions are sorted by decreasing standard deviation. To minimize compute time, clips that are very close to an existing tree node are held at that node, and not pushed down into the tree.

Both Shape Clustering and Shape Classification require a distance metric between two images or two cluster means. The metric is based on template matching, but a simple sum of squares of greyscale differences is inadequate, since edges of strokes may not be perfectly aligned, and yet it is desirable to distinguish 'i' from 'í'. The distance metric penalizes differences that occur in areas with a low gradient, and with one of the greyscale values near black or white.

B. *Shape Classification*

All clusters are initially labeled as type *Master*. All clusters are then compared to all other clusters of the *same* class label, that have more samples, and if the distance is less than a threshold, the type of the smaller cluster is changed to *Dependent* (of the Master). The Dependent clusters then follow the fate of their assigned Master.

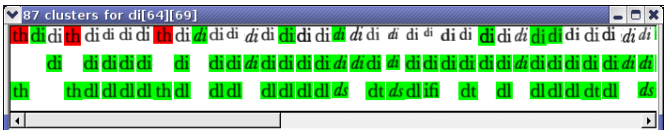

Figure 2. Results of shape cluster calssification. *Master*, *Reject* and *Dependent* clusters are colored green, red, transparent, respectively.

Each Master cluster is now compared to each Master cluster with more samples, but with a *different* class label. If the distance is less than a threshold, then the smaller cluster is re-typed as *Reject*, and will be subject to possible correction. Fig. 2 shows the result of this operation. The top row shows the clusters of 'di' sorted by frequency, and colored green for Master, red for Reject, and no color for Dependent. The second row shows the nearest Master of 'di' for the cluster directly above, and the third row shows the nearest Master of class other than 'di' The three clusters of 'di' that are actually 'th' are marked Reject.

Note with the inclusion of bigrams, correction of most segmentation errors is achieved without having to reconsider the character segmentation everywhere. Also this 'di'->'th' correction is obvious with bigrams due to the match with another bigram. With only unigrams, there would not be a match to either of the 'd' or 'i' individually. With bigrams, on the other hand, even quite complex segmentation errors can be corrected, as shown in Fig.3.

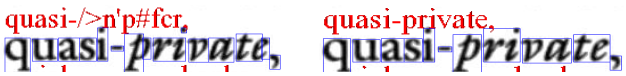

Figure 3. An example of before-and-after image-based correction.

The inclusion of bigrams also introduces a new problem. Bigrams and unigrams may disagree over whether a character should be corrected. Such disagreements are resolved by allowing the bigrams to overrule the unigram cluster type. For each Master unigram cluster, if a sufficient proportion of the samples are included in a Reject bigram cluster, then that unigram cluster is also marked Reject. Conversely if the majority of samples in a Reject unigram clusters are in a Master bigram cluster, then the unigram cluster in re-typed as Master.

C. *Hypotheses Verification*

When the language-based correction path generates a hypothesis, it needs to be validated by the image model before the change is accepted. The change is first decomposed into a set of atomic operations such as symbol insertion/deletion, case-folding, space insertion/deletion, or substitution. For example, a correction from "*fmanCial*" to "*financial*" would result in a substitution {*m*→*in*} and a case-folding {*C*→*c*}. A substitution change is acceptable if the sample *m* is from a small cluster or if its distance to the nearest *in* cluster is below a threshold. To verify a case-folding change, we use the bounding box profile relative to its neighboring symbols and the cluster distance between C and c. Similar rules are defined for each type of modification. A change is accepted if none of its constituent operations is rejected. It is not the goal to find a perfect solution for reconciling confidence scores between these two models, but simply to reject unlikely changes based on image features.

IV. Language-Based Correction

The organization and functionality of the language correction path is analogous to the image path. Treating normalized word tokens as a cluster, all occurrences of the same token are evaluated together in context in an attempt to find any alternatives that would result in higher likelihoods. If an alternative is found, each instance is then individually evaluated against both the language and image model to determined if the correction should be accepted.

A. *Adaptive Language Model*

Our base language model consists of word n-grams with backoff to character n-grams for out-of-vocabulary terms trained on a large web corpus. Since a book typically contains tens of thousands of words with a significantly different distribution, especially on content keywords such as proper nouns or topic indicators, it is important to augment with an adaptive cache model.

Considering that the goal is to remedy the low prior probability of content keywords in the base model, we used a simple word frequency table as the cache model. The model is initialized to be empty, and words are added to the model by one of three criteria. The first source is the OCR engine. If the word is marked as a dictionary word by the engine or has really high confidence, it is accepted into the model. The second source is the base language model. Any word that is verified by the base word model (with no correction hypothesis) is added. To address the situation of content keywords which are not in the base model, we also include high frequency OOV (out-of-vocabulary) words which have no viable alternative from the language model and have high confidence. We compute the frequencies of these words relative to the total number of words in the book.

There are several ways to combine the base and cache model, such as weighted average or maximum entropy. We found simply taking the max worked better than other more complicated weighting schemes. If a token has an entry in the cache model with a higher prior, this overrides the the base model probability $P(w)=\max\{P_{\text{cache}}(w), P_{\text{base}}(w)\}$. Although this does not correctly adjust the conditional probabilities when we

evaluate likelihood of a bigger context containing w, adjusting the prior alone seems to be quite effective.

B. Hypotheses Generation

In order to identify potential errors, we would like to isolate words or segments which have low likelihood according to our model. In addition, we also want to generate a correction candidate that is more likely than the current label. In order to do this effectively, we consider all occurrences of each OOV word w in their respective context jointly. More precisely, let $C_i(w)$ be a window of n=7 words centered at the ith instance of w, we want to find the most likely word label q given the recognition output w and its context $C_i(w)$,

$$q_i^* = \text{argmax}_q\, P(q|w,C_i(w))$$
$$\sim\text{argmax}_q\, \log P(w|q,C_i(w))+\log P(q,C_i(w))$$

where $P(q,C_i(w))$ is the language model likelihood for a given context, and $P(w|q)$ is the noise channel embodying confusion probabilities[4], which we assume to be independent of neighboring words. Using Viterbi search, we could find the most likely candidate q_i^* in a given context. If $q_i^*=w$, it means there is no better answer than current label. Repeating the process for each context would produce candidates $q_1^*..q_N^*$.

Unfortunately, since our cache model is not integrated with the base model, neither the calculation of the likelihood $P(q,C_i(w))$ nor the Viterbi search would correctly account for the adjustment. Consequently, the model would unfairly penalize a word that has a higher prior than in the base model, and would be less likely to find the word as the top alternative during search. We approximate this using a log-linear formulation, where the likelihoods are essentially computed separately then summed together.

$$q_i^* = \text{argmax}_q\, \{\, \alpha\, (\log P_{\text{base}}(w|q)+\log P_{\text{base}}(q,C_i(w))) +$$
$$\beta\, (\log P_{\text{cache}}(w|q)+\log P_{\text{cache}}(q))\, \}$$

where $P_{\text{cache}}(q)$ is the document frequency described above, and $P_{\text{cache}}(w|q)$ is a static OCR confusion table augmented with verified document corrections. No attempt is made to learn α and β, which are set to 1.

A simple strategy is to simply replace w with q_i^* for each instance independently. However, this approach often misses a correction for lack of sufficient evidence in short or ambiguous contexts. Therefore, we sum the likelihood ratio over all contexts to derive the most likely answer and apply the same correction to all instances.

This is illustrated in Fig.4. We evaluate the word token "thinx" in four different contexts. Suppose Viterbi search produced "think" twice and "thank" once as better alternatives, and made no suggestion in the other. The likelihood ratio of each candidate to the base hypothesis is summed over all contexts and remapped to a confidence to produce the top choice "think". To prevent the situation where the correct answer for separate instances are indeed different, as in the last example of "thank you very much", we check the likelihood $P(q^*,C_i(w))$ against $P(w,C_i(w))$ for each instance, and generate a correction hypothesis only if the likelihood improves.

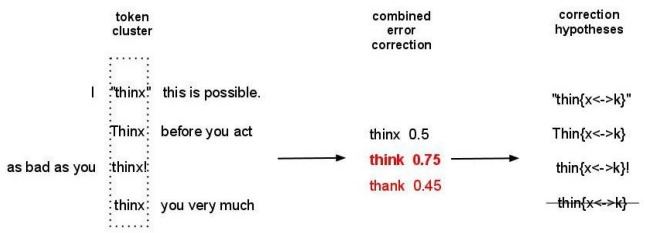

Figure 4. Combined error correction.

C. Case Inference

One practical issue worth discussing is token normalization. Although it is possible to absorb all case variants, punctuation and ligatures into a single model, these variations are typically normalized out in a language model. Tokens are preprocessed entering the model, and corrections denormalized on return. Unfortunately, Tesseract makes a fair number of casing errors, usually due to mis-adaptation on similar shape pairs such as c/C, w/W. Using information such as sentence boundary, word context, capitalization rules, one could infer the most likely surface form of a given word as described in TrueCase[6]. Although this provides a good prior for word casing using only textual information, it does not account for the strong signals derived from the image. For example, words like "DVD" maybe capitalized or not in identical context. Using probability computed from textual information alone will force the same surface form on all occasions and inevitably make many errors.

The strongest signal comes from the image, which is indirectly observed through the initial answer provided by the OCR engine. We added a feature based on edit distance cost between raw OCR output to each surface form. Another feature is based on comparing the expected word shape of a surface form to the actual bounding box profile. Since book headers are often capitalized or in all-caps, a different prior distribution is estimated for header blocks.

Using the same approach used to generate hypotheses for a word given its context, we learned a model to infer the most likely casing using these signals. After word spelling has been corrected, we apply the case inference as part of the denormalization process on each instance.

V. Evaluation and Results

The system was evaluated on scanned books, similar in quality to those in the Google1000 dataset [9]. A *Smoke* set consisting of roughly 2 million words (11 million chars) was used during development for error analysis and system tuning. Final testing was done on a larger set *Test* of 6 million words. The base language model was trained on a web corpus and the noise channel was learned from spelling errors. For the case inference model, we collected statistics from the Google Ngram Viewer data [12] and the geometrical features were tuned on the smoke set.

Before moving on to experimental results, we should explain the evaluation process and metrics. Since it is difficult to manually collect and verify a dataset of this size, we rely on a semi-automatic approach by aligning scanned books with available PDF sources. Several factors contribute to the imperfection of this ground truth data, including duplicate or

shuffled pages during scanning, different block ordering in the output serialization, mismatch in editions between PDF and scanned versions, etc. Therefore, we have developed an automatic evaluation method based on string alignment. Consequently, variations exist across experiments due to differences in aligned segments.

A large set of metrics are defined to help measure various aspects of the results. Three of the most important metrics are summarized here. The *ch.subst* measures the character substitution error rate in the aligned section of text. This measure is case and punctuation sensitive. The *wd.err* rate is based on case-folded tokens stripped of punctuation marks and excludes stopwords, but includes word substitution, insertion and deletion. The measure makes sense from an indexing and search perspective, but is more susceptible to alignment difference. To mitigate the effect due to alignment, the *flwd.drop* rate considers the ground truth as a bag of words, and computes the percentage of ground truth words that are missing from OCR output.

The results are summarized in Table 1. Using Tesseract output as a baseline, the columns show the percent relative change in each of the metrics obtained by the proposed method *LIM*. For reference, improvements yielded using only the image-based correction (*IM*) and only language-based correction (*LM*) are also given.

On the *smoke* set, the ch.subst, wd.err and flwd.drop rate decreased by 36.88%, 22.58% and 24.39%, respectively. The results on IM and LM showed both correction branches contributed substantially. The additional improvements in character substitution rate are mostly attributed to fixing case errors and multiple errors in the same word. On *Test*, word error rate is reduced by 18%. Both IM and LM generated less reduction on this larger set. The ch.subst, however, showed a much smaller reduction of 13%. The much smaller 4.85% reduction from LM in ch.subst relative to the 12% word error reduction suggests that the case inference model generalized poorly on this set.

TABLE I. SYSTEM PERFORMANCE IMPROVEMENT OVER BASELINE TESSERACT.

Dataset	model	Δch.subst%	Δwd.err%	Δflwd.drop%
Smoke	IM	-7.92	-7.5	-7.47
	LM	-30.47	-17.74	-19.23
	LIM	**-36.88**	**-22.58**	**-24.39**
Test	IM	-6.77	-3.19	-3.14
	LM	-4.85	-10.56	-12.04
	LIM	**-13.58**	**-16.11**	**-18.16**

A more detailed analysis showed that the proposed solution improved on almost every book, except for a book on statistical physics and protein folding where the case inference model tried too hard to correct poorly recognized equations containing mixed-cased words and ended up making more substitution errors. However, the overall flwd.drop rate still improved, meaning misspelled words were corrected. Another failure case occurred when an accented foreign name was consistently recognized as a more common English name. This suggests that the adaptive model needs to be refined and image model verification needs to be tightened.

We measure runtime as a percent of total CPU time taken by the base Tesseract on the same dataset. Since the adaptive image model needs to cluster and classify all unigram and bigram segments, runtime varied greatly depending on image quality and page content, measuring 55% on Smoke and 84% on Test. It is more difficult to measure the runtime for the language correction path since it employs a network-distributed service. We estimate the total language CPU overhead is 10-15% on top of OCR.

VI. CONCLUSIONS

We presented a system which consists of two correction paths based on document-specific image and language models. Each adaptive model exploits the redundancy in fonts and vocabularies to detect inconsistencies, but leverages the orthogonality of the other to verify correction hypotheses. The system was able to reduce the word error rate of Tesseract by 25% on a large test set.

Overall, we are very encouraged by the results considering the numerous simplification steps taken in the system. We believe that substantial improvements can be made by addressing some of those issues. For example, the base language correction model was trained on a web corpus using query spelling errors as the noise channel; it clearly should be retrained using a book corpus and OCR confusions. Similarly, a better integration of the document language model into the Viterbi search process should generate better corrections. Furthermore, minimum error training could be applied to fine-tuning system parameters such as image model verification thresholds and log-linear coefficients. The same strategy, excluding the case inference model, has been tried on other Latin-family languages with similar success. Generalizing the approach to non-alphabetical or non-word-based languages remains as future work.

REFERENCES

[1] T.K. Ho, G. Nagy, "OCR with no shape training," ICPR, pp.27-30, 2000.

[2] A. Kae, G. Huang, C. Doersch, E. Learned-Miller, "Improving state-of-the-art OCR through high-precision document-specific modeling," CVPR, pp.1935-1942, 2010.

[3] A. Kae, E. Learned-Miller, "Learning on the Fly: Font-Free Approaches to Difficult OCR Problems," ICDAR 2009.

[4] O. Kolak, P. Resnik, "OCR error correction using a noisy channel model," Human Language Technology Conf., 2002.

[5] C.J. Leggetter, P.C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," Computer Speech & Language, 9(2), pp171-185, 1995.

[6] L.V. Lita, A. Ittycheriah, S. Roukos, N. Kambhatla, "tRuEcasIng," ACL, 2003.

[7] P. Sarkar, G. Nagy, "Style consistent classification of isogenous patterns," IEEE Tans. on PAMI, 27(1), January, 2005.

[8] R. Smith, "An overview of the Tesseract OCR engine," ICDAR, 2007.

[9] L. Vincent, "Google book search: document understanding on a massive scale," ICDAR, 2007.

[10] S. Veeramachaneni, G. Nagy, "Adaptive classifiers for multi-source OCR," IJDAR, vol.6, pp.154-166, 2003.

[11] P Xiu, H Baird, "Towards whole book recognition," DAS, pp.629-636, 2008.

[12] http://ngrams.googlelabs.com/datasets