2009 10th International Conference on Document Analysis and Recognition

Low Cost Correction of OCR Errors Using Learning in a Multi-Engine Environment

Ahmad Abdulkader Matthew R. Casey
Google Inc.
ahmad@abdulkader.org *mrcasey@google.com*

Abstract

We propose a low cost method for the correction of the output of OCR engines through the use of human labor. The method employs an error estimator neural network that learns to assess the error probability of every word from ground-truth data. The error estimator uses features computed from the outputs of multiple OCR engines. The output probability error estimate is used to decide which words are inspected by humans. The error estimator is trained to optimize the area under the word error ROC leading to an improved efficiency of the human correction process. A significant reduction in cost is achieved by clustering similar words together during the correction process. We also show how active learning techniques are used to further improve the efficiency of the error estimator.

Keywords: {OCR Correction, Multiple Engines, Machine Learning, Clustering, Active Learning}

1. Introduction

It is estimated that there are between 50 and 200 million books ever published [9]. A significant fraction of these books are only available in non-digital forms. Furthermore, most of the estimated 100,000 new books published yearly are published exclusively on paper for many reasons including copyright [10]. This prohibits the automatic indexing and searching of these documents and consequently limits their availability and accessibility on digital medium like the Internet, PDA and mobile phones.

Optical Character Recognition (OCR) technology has been used for decades to convert scanned images of documents to editable text. The accuracy of commercially available OCR engines has been constantly improving to the extent that the OCR problem has been regarded by many as a solved one. However, in practice, it has been found that the mean word level error rates for OCR ranges roughly between 1 to 10%. The determining factors for OCR accuracy are the quality, age and structure complexity of the documents as well as the artifacts introduced during the scanning process. This accuracy range is generally adequate for many applications of OCR. Nevertheless, it is not adequate for a slew of other applications like information retrieval (IR). A number of large IR projects rely on OCR as the main method for data acquisition. Examples of these projects are Google's Book Search [9], the Gutenberg project [1] and the Million Book Digital Library Project [2]. Other applications that require low OCR error rates are text-to-speech and repurposing.

One possible way to improve accuracy is to use human labor to correct OCR errors. This process can be confusing, time consuming and costly to the extent that it might seem cheaper to retype a document from scratch rather than relying on OCR and correcting its errors. In this work, we seek a low cost method that uses human labor efficiently for correcting OCR errors. Our goal is to reach a word error rate of around 0.5% at the minimal possible cost of time and money.

It has long been observed that different classifiers make different errors. The theoretical framework, results and methods of combing multiple classifiers to improve accuracy have been well studied in the pattern recognition and machine learning literature [6]. Similar ideas have been proposed to automatically post process OCR results [10]. Different OCR engines also tend to make different mistakes. This observation can be explained by the fact that different engines use different approaches for classifying characters, are possibly trained on different data sets with different character and word distributions and are possibly using different language models.

Our approach is based on a related concept. We observe that the errors made by a primary OCR engine are highly correlated with its disagreements with subsequent secondary OCR engines. We use this correlation to predict the existence of errors and to prioritize them according to an error estimate.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.242

IEEE computer society

In the following sections, the main approach will be described in detail. Further extensions that significantly improve the efficiency of the error estimation will be described, as well as experiments and results.

2. The Proposed Algorithm

In this approach, a state-of-the-art commercial OCR engine is used as a primary engine. We also employ a number (n) of secondary weaker OCR engines. As mentioned in the introduction, we base our algorithm on the simple observation that the recognition errors made by the primary engine are highly correlated with its disagreements with the secondary OCR engines. On the contrary, the agreement of all (or most) engines indicates a very strong confidence. To estimate the existence of an error, we use an error estimator that operates on the outputs of all the OCR engines' outputs as features. A block diagram of the proposed algorithm is shown in Figure 1. It goes as follows:

1. The input set of documents is fed to the primary and the secondary OCR engines.
2. In an error estimation phase, the outputs of the OCR engines are examined. Words for which all (or almost all) OCR results are equal are accepted as correct. Words whose OCR results differ are subjected to the error estimator. The features of the error estimator are computed from the results of all engines. Words whose error estimate exceeds a pre-defined threshold α are considered "suspects". Suspects for all documents are generated. The detailed structure of the error estimator, its input features and how it is trained are detailed in sections 3 & 7.
3. In a clustering phase, suspects with equal primary OCR results and whose word images closely match are grouped together in clusters. The median suspect from each cluster is picked as a representative of the whole cluster. As a result of clustering, the number of suspects is dramatically reduced. The details of the clustering phase and its impact on the efficiency of the system are described in sections 4 & 7.
4. The resulting suspects after clustering are then sent to a manual Quality Assurance (QA) system. Human operators are shown images of individual words in context. The operators are asked to pick an answer from the list of possible answers from the different OCR engines or type a new answer. More details about the QA user interface are provided in section 5.
5. Based on the answers of the human operators, corrected OCR documents are then generated.

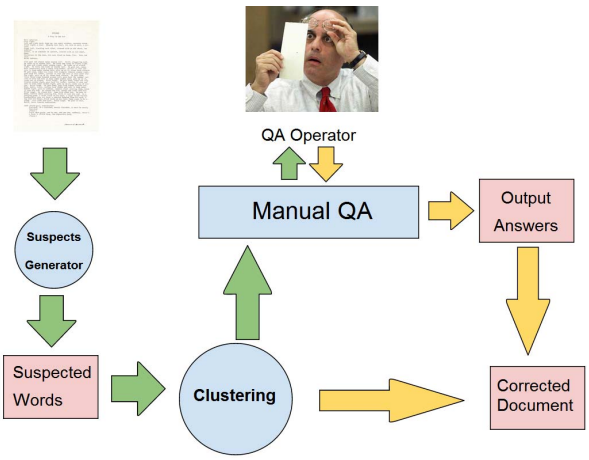

Figure 1. A block diagram of the proposed algorithm

It should be noted that the fraction of the words to which the error estimator is subjected varies greatly with the quality of the document. Table 1 shows the agreement and disagreement rates for two different data sets in a two-OCR-engine correction system.

Dataset Quality	*Primary Error Rate*	*Disagreement Rate*	*Agree & Wrong*	*Disagree & Wrong*
High	0.80%	4.00%	0.15%	0.32% (8% of disagreements)
Low	4.00%	16.00%	0.50%	4% (25% of disagreements)

Table 1. Agreement & Disagreement rates for documents of different quality

3. The Error Estimator

The error estimator is a classifier that predicts whether the OCR result of a specific word from the primary engine is an error based on the input features of this word. The classifier used in our approach is an artificial neural network. The input features the fed into the network are described below:

1. The confidences values of all OCR engines
2. String lengths of all OCR results.
3. Mutual confidences of primary and secondary OCR results. This assumes that the OCR engines are capable of computing a confidence for arbitrary words. This is possible for most engines, but if this capability is missing, the corresponding features can be omitted.
4. Mutual ranks of primary and secondary OCR results. Most engines produce a list of alternate answers. This subset of features represents the position of each OCR result in the lists proposed by the other engines.
5. A binary feature for each OCR result denoting whether it is a valid dictionary word.

6. An n-gram character model score computed for each of the OCR results.

As shown in Table 1, it was observed that the different engines disagree between 4 and 16% depending on the quality of the document. Moreover, the error rate on the fraction of the words for which the OCR engines agree ranges between 0.15 and 0.5%. We will refer to this set of errors as residual errors. Based on these statistics, it was concluded that it would be more efficient to run the error estimator only when the OCR engines disagree. Further investigation of the residual errors indicated that they are mostly errors in the ground truth or highly ambiguous cases.

The neural network built in our experiments has a single hidden layer with 8 nodes. The architecture has been evolved empirically during experimentation. The net is trained on a ground truth data set that contains around 200,000 samples. The prior probability of errors in this data set is 12.25%. A smaller set of a similar distribution is used as a validation set. The net has been trained to optimize the area under the Receiver Operating Characteristic (ROC) graph [3].

Figure 2 shows the relative error reduction rate against the relative rejection rate. It is also shown how the error estimator significantly improves the error reduction rate compared to the confidence level of the primary OCR engine.

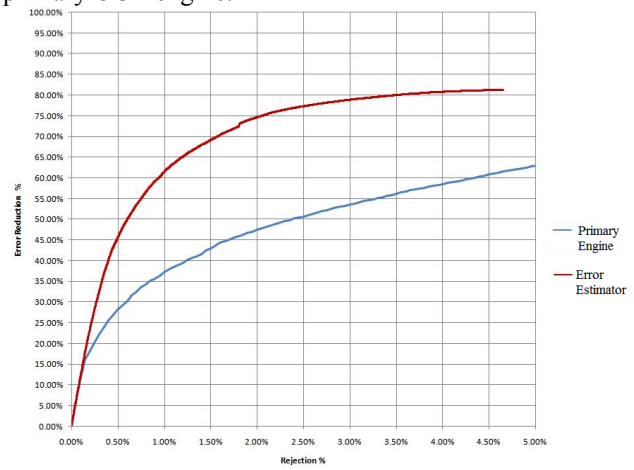

Figure 2. The ROC of the estimator compared to that of the primary OCR engine

4. Clustering Suspects

When performing OCR correction for a set of related documents or books, it was observed that many of the suspects generated by the error estimator are almost identical. This is not entirely surprising given that OCR engines perform consistently when subjected to similar inputs. It was then concluded that the number of suspects can be dramatically reduced if similar suspects that will have the same answer were clustered together and treated as a single suspect.

For this purpose, a simple, crude, and effective method for dramatically reducing the number of suspects was devised. For a given prospective suspect, we want to determine if an equivalent matching suspect (one probably with the same answer) had already been generated. The matching suspect may or may not have been processed by a QA operator already.

To do this, the image bitmap of the prospective suspect is matched against all the previously generated suspects. A number of cross correlation based template matching algorithms [8] can be used for this purpose. This scheme would find good matches for a prospective suspect but has the serious drawback of being of $O(n^2)$ with the number of suspects. It would quickly slow down as the number of suspects increase. In order to limit the search space, the suspects search was limited to the suspects that have the same primary OCR string as the prospective suspect. Although the matching process is still of order $O(m^2)$, the number of suspects with matching OCR primary strings m is much smaller than the total number of suspects n. As a result, the matching process is much faster. The suspects clustering algorithm goes as follows:

1. Create an empty list of suspect clusters C.
2. Given a prospective suspect word P whose error estimate exceeds the pre-determined threshold α, find the set S, where $S \subset C$, whose member clusters all have the same primary OCR string as P.
3. Find the cluster M, where $M \in S$, that has the minimum possible template-matching distance d to P.
4. If $d \geq$ a pre-defined threshold β, a new cluster N is created and added to C. In addition, a new question corresponding to N is generated and queued to be answered by a QA operator.
5. If $d < \beta$, P is added to the cluster M. Because M is an already existing cluster, a question must have been generated before for it. If an answer had already been provided to cluster M, P is automatically answered.
6. When a QA operator answers a question for a specific suspect's cluster, the answer is propagated to all the suspects in the cluster.

As mentioned earlier, suspect clustering results in significant improvements to the efficiency of the overall correction system. More details are provided in section 7. The performance of the clustering algorithm can be measured in isolation by looking at the mean hit rate of the clusters as a time series. The performance varies significantly depending on the quality of the

documents being processed. We have observed, as a rule of thumb for most documents, that clustering cuts the cost of QA in half. We have also observed a peak cost reduction of 80% in some sets of documents.

Figure 3 shows the mean cluster hit rate as a time series. Each point represents the mean hit rate over 200 consecutively-processed sets of documents. It is worth noting that the sudden drop in the mean hit rate shown in the graph is associated with the transition from good to bad quality scans. The hit rate drops because of noise in the image causing the image matching distance threshold to be exceeded as well as a decreased stability of the primary OCR output. This is not surprising given that the proposed clustering scheme is based upon the assumption that the primary OCR results will be consistent across different pages. This assumption might hold less for lower quality documents.

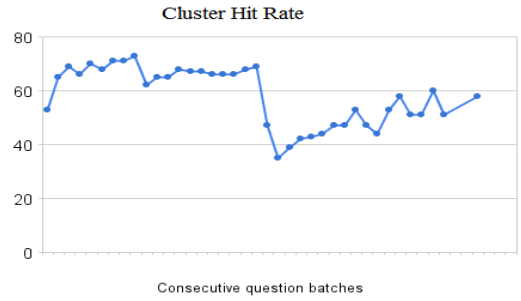

Figure 3. The mean hit rate of the suspects' clusters as a time series

5. The User Interface

The user interface presented to the QA operators plays a key role in the cost reduction. Figure 4 shows a snapshot of the UI used in our system.

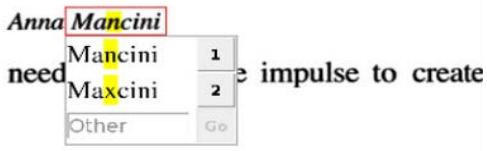

Figure 4. A snap shot of the user interface presented to the QA operators

As shown in the figure, the following aspects of the user interface should be noted:

1. The operator is presented with suspected word in context. The image of the suspected word is highlighted using a bounding box.
2. The user is presented with several possible choices for the correct answer of the suspected word. These possible answers are extracted from the outputs of all the OCR engines used in the system. The operator can then choose any of these answers using a single keystroke (1, 2, etc…). In the event that the correct answer is not in the list, the QA operator needs to type it. Based on data gathered from the instrumentation of the user interface it has been observed that the operators take under 3 seconds to respond to a question if the answer is one of the listed choices, while it takes around 6.5 seconds to type in an answer. Based on the statistics gathered from the data sets that were used in evaluation, it is estimated that the typing option is only used between 8-33% of the time. On average it takes under 4 seconds for an operator to respond to a question.
3. Using a simple string edit distance algorithm, the differences between the different possible answers are highlighted as well as the corresponding characters in the word image.

It is believed that the aspects highlighted above significantly reduce the cognitive load to which a QA operator is subjected, resulting in a significant reduction of the overall correction cost.

6. Active Learning

Active learning methods [4] are typically used in situations in which unlabeled data is abundant but labeling data is expensive. In such a scenario the learning algorithm can actively query the user/teacher for labels. We have attempted to use the same principle in tuning the error estimator for a specific task.

While processing a large collection of documents, a relatively large set of suspects is generated. The set of suspects is sorted in descending order based on the error estimate. The set is then divided to smaller batches that are provided to the QA operators in order, one batch at a time. Once the first batch of questions is fully processed by QA operators, it is regarded as labeled data. This labeled data is used to tune, or further train, the error estimator. The newly tuned error estimator is used to estimate the error on the subsequent batches. This technique further reduces the cost of the error correction process. More details are provided in section 7.

7. Experiments and Results

In our experiments, we were seeking to evaluate the cost of acquiring textual content from a large set of books using the proposed method and compare its cost to the cost of typing the same set of books using professional typists who can attain a peak typing speed of 75 words per minute (WPM) at a word error rate of roughly 0.5% [5].

As mentioned in section 3, we trained an error estimator on a set of roughly 200,000 suspects that

were extracted from a set of labeled documents. We then proceeded to test the system on two different sets of books. The first set, Set1, has around 1,300,000 words and a primary OCR word error rate of 2.43%. The second set, Set2, has around 800,000 words and a primary OCR word error rate of around 5.23%. Set2 consists of older books with more significant degradation. None of these sets were used to train the error estimator. The output of the OCR was processed through three different configurations of our system. The first configuration used only the error estimator. The second configuration used the error estimator and the cluster algorithm described in section 4. The third configuration added active learning as described in section 6. The output of the system was then assessed by human operators to compute the word error rates. Figure 5 shows how the word error rates on both sets changed with number of questions per word (QPW). QPW is used here as a proxy for cost.

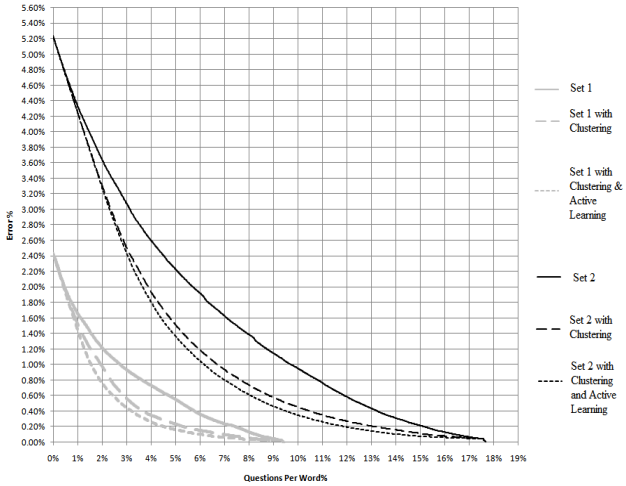

Figure 5. The characteristics of three different configurations of correction system on two sets of books

From the figure, we observe that the target error rate can be reached at roughly a QPW of 8.5% for Set1 and a QPW of 2.7% for Set2. From section 5 we recall that the average time to respond to a question is 5 seconds. Accordingly, the effective throughput of the system is around 151 WPM for Set1 and 444 WPM for Set2 corresponding to a cost reduction of 50% to 83% depending on the quality of the documents.

8. Conclusions

The system described in this paper provides an efficient low cost method for digitizing textual context. Our system provides significant reductions in cost over a typist based system. This is achieved by using learning methods to estimate OCR errors, using clustering to group similar errors, designing a user interface that minimized the cognitive load on QA operators and using active learning techniques to tune the error estimation on the targeted document collection.

9. References

[1] "Project Gutenberg", http://www.gutenberg.org

[2] "The Million Book Digital Library Project", http://www.rr.cs.cmu.edu/mbdl.htm

[3] Cortes, C. and Mohri, M, "AUC Optimization vs. Error Rate Minimization", *Neural Information Processing Systems (NIPS), MIT Press*, 2004.

[4] F. Provost, M. Saar-Tsechansky, "Active learning for class probability estimation and ranking". *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann*, 2001, pp. 911–920.

[5] F. Vorbeck, A.Ba-Salamah, J. Kettenbach and P. Huebsch, "Report generation using digital speech recognition in radiology", Springer-Verlag, 2000.

[6] J. Kittler, M. Hatef, R. P. Duin and J. G. Matas, "On combining classifiers", *IEEE Transactions on PAMI 20 (3)* (1998), pp 226-239.

[7] J. L. Crowley and J. Martin, "Experimental comparison of correlation techniques", *IAS-4, International Conference on Intelligent Autonomous Systems,* March 1995.

[8] Jurie, F. and Dhome, M., "A simple and efficient template matching algorithm", *Proceedings of the International Conference on Computer Vision,* 2001.

[9] L. Vincent, "Google Book Search: Document Understanding on a Massive Scale", *Proceedings. Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 2007, pp. 819-823.

[10] S. T. Klein and M. Kopel. "A voting system for automatic OCR correction". *Proceedings of the SIGIR 2002 Workshop on Information Retrieval and OCR*, August 2002.