Journal of Machine Learning Research 13 (2012) 363-387 Submitted 5/10; Revised 8/11; Published 2/12

Bounding the Probability of Error for High Precision Optical Character Recognition

Gary B. Huang GBHUANG@CS.UMASS.EDU
Andrew Kae AKAE@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003

Carl Doersch CDOERSCH@CS.CMU.EDU
Department of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Erik Learned-Miller ELM@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts Amherst
140 Governors Drive
Amherst, MA 01003

Editor: Leon Bottou

Abstract

We consider a model for which it is important, early in processing, to estimate some variables with high precision, but perhaps at relatively low recall. If some variables can be identified with near certainty, they can be conditioned upon, allowing further inference to be done efficiently. Specifically, we consider optical character recognition (OCR) systems that can be bootstrapped by identifying a subset of correctly translated document words with very high precision. This "clean set" is subsequently used as document-specific training data. While OCR systems produce confidence measures for the identity of each letter or word, thresholding these values still produces a significant number of errors.

We introduce a novel technique for identifying a set of correct words with very high precision. Rather than estimating posterior probabilities, we **bound** the probability that any given word is incorrect using an approximate worst case analysis. We give empirical results on a data set of difficult historical newspaper scans, demonstrating that our method for identifying correct words makes only two errors in 56 documents. Using document-specific character models generated from this data, we are able to reduce the error over properly segmented characters by 34.1% from an initial OCR system's translation.[1]

Keywords: optical character recognition, probability bounding, document-specific modeling, computer vision

1. This work is an expanded and revised version of Kae et al. (2010). Supported by NSF Grant IIS-0916555.

1. Introduction

A long-standing desire in classification has been the ability to adapt a model specifically to a given test instance. For instance, if one had a reliable method for gauging the probability of correctness of an initial set of predictions, then one could iteratively use the predictions likely to be correct to refine the classification model and adapt to the specific test distribution. This general strategy has been considered in the past (Ho, 1998; Hong and Hull, 1995b,c), but particularly within the domain of computer vision, it has not had much success. We believe that this lack of success stems from the difficulty of reliably estimating probabilities in the high dimensional vector spaces common in computer vision. Rather than attempting to reliably estimate probabilities for all predictions, we instead propose a shift in perspective, focusing on identifying cases where we can reliably bound probabilities.

We show that this old idea of using a first pass system to identify some reliable samples, which are then used in turn to train a second pass system, can be quite powerful when the method of selecting reliable samples is appropriate. In particular, by formally upper bounding the probability of error in a first pass system, we can select results whose probability of error is not greater than some very small threshold, leading to the automatic selection of a subset of results of the first pass system with very low error rate. These results can be considered highly reliable "training data", specific to the test distribution, for a second pass system. Using this test-specific training data, we demonstrate significant error reductions on some difficult OCR problems. Thus, the main contribution of our paper is the combination of standard bounding techniques with the idea of multi-pass test-specific classification systems. To our knowledge, there is no preceding work which does this.

We first describe why adapting a model to a specific test distribution is an important goal, and in Section 2, discuss our rationale for bounding rather than estimating probabilities.

1.1 Adapting to the Test Distribution

In supervised learning, we are given training data $\{(x_i, y_i)\}$ to learn a model capable of predicting variables y from observations x, and apply this model at test time to new, previously unseen observations x'. An important implicit assumption in this framework is that the training instances (x_i, y_i) are drawn from the same distribution as the test instances (x', y'). Unfortunately, however, this is often not the case, and when this assumption is violated, the performance of supervised learning techniques can decay rapidly.

One natural setting in which this scenario arises is text recognition. In everyday life, we encounter a variety of fonts and character appearances that differ widely from each other and may be entirely new to us, such as in outdoor signs, graffiti, and handwritten messages. Despite not having appropriate labeled training examples, as humans we would be able to quickly adapt and recognize such text, whereas a machine learning algorithm would not.

There are several methods of addressing this problem. We may attempt to leverage knowledge from a closely related task and apply that knowledge to solve the new test problem, as in transfer learning. Alternatively, we may attempt to explicitly parameterize and model the manner in which the data varies, as in a hierarchical Bayes model. Instead, we argue for a third, non-parametric option, inspired by human behavior.

When presented with text in an unusual font, or with a new situation in general, we argue that humans will first identify elements that they are very confident in their understanding of, based on

previous experience. For instance, they may be able to identify a particular letter based on similarity to previously seen fonts or by the occurrence statistics. Once they have done this, they will condition on this information and use it as an aid to understanding the remaining elements.

Similarly, we argue that a machine learning algorithm could benefit by first understanding, with very low probability of being incorrect, some subset of the new test instance, and conditioning on this information as training data specific to the test case.

Ideally, rather than making a hard decision and potentially throwing away useful information, we would like to maintain a distribution over the possible interpretations, such as a distribution over the possible characters a particular letter could be. We could then reason probabilistically over the different joint labelings of all characters to determine a maximum a posteriori estimate. In practice, however, we believe this has two pitfalls. The first is the difficulty in obtaining accurate distributions over labels, especially when dealing with very high dimensional data, as we describe later. These initial errors can then propagate as we do further reasoning. The second is the computational complexity of performing learning and inference on such distributions over labelings. Instead, by making a hard decision, we can make learning and inference much more efficient, and make use of the conditioned information as test-case specific training data with minor modifications to standard algorithms. In essence, by making hard decisions only where we are very confident of the labeling, we gain the computational efficiencies associated with making such decisions without the common risk of making unrecoverable errors.

1.2 Document-Specific OCR

In this paper, we focus on the problem of improving optical character recognition (OCR) performance on difficult test cases. In these instances, the non-stationarity between the distribution in the training examples and distribution in the test cases arises due to factors such as non-standard fonts and corruption from noise and low resolution.

Applying the reasoning above, we would like to obtain training data from the test documents themselves. In this paper, we use the output from an OCR program and identify a list of words which the program got correct. We can then use these correct words to build new, *document-specific* OCR models.

While identifying correct words in OCR program output may seem like an easy thing to do, to our knowledge, there are no existing techniques to perform this task with very high accuracy. There are many methods that could be used to produce lists of words that are mostly correct, but contain some errors. Unfortunately, such lists are not much good as training data for document-specific models since they contain errors, and these errors in training propagate to create more errors later.

Although some classifiers may be robust to errors in the training data, this will be very dependent on the number of training examples available. For characters such as 'j' that appear less frequently, having even a few errors may mean that more than half of the training examples are incorrect. While we can tolerate some errors in character sets such as 'e', we cannot tolerate them everywhere.

Thus, it is essential that our error rate be very low in the list of words we choose as correct. As described below, our error rate is less than 0.002, as predicted by our theoretical bounds, making our generated lists appropriate for training document-specific models.

We first give some background on why we believe this problem of bounding probabilities to achieve high-precision, document-specific training data is interesting. In Section 3, we present the specifics of our method for creating nearly error-free training sets, and give theoretical bounds on the

probability of error in these sets in Section 4. We then describe how we use the document-specific model to reduce the error rate in Section 5, and give experimental set-up and results in Section 6. Finally, we conclude with directions for future research in OCR, as well as potential applications of our method to other problem domains, in Section 8.

2. Background

Humans and machines both make lots of errors in recognition problems. However, one of the most interesting differences between people and machines is that, for some inputs, humans are extremely confident of their results and appear to be well-justified in this confidence. Machines, on the other hand, while producing numbers such as posterior probabilities, which are supposed to represent confidences, are often wrong even when posterior probabilities are extremely close to 1.

This is a particularly vexing problem when using generative models in areas like computer vision and pattern recognition. For example, consider a two class problem in which we are discriminating between two similar image classes, A and B. Because images are so high-dimensional, likelihood exponents are frequently very small, and small percentage errors in these exponents can render the posteriors meaningless. For example, suppose that $Pr(\texttt{image}|\texttt{A}) = \exp(-1000 + \varepsilon_{\texttt{A}})$ and $Pr(\texttt{image}|\texttt{B}) = \exp(-1005 + \varepsilon_{\texttt{B}})$, where ε_A and ε_B represent errors in the estimates of the image distributions.[2] Assuming a roughly equal prior on A and B, if ε_A and ε_B are Gaussian distributed with standard deviation a small proportion (for instance, around 1%) of the magnitude of the exponents, the estimate of the posterior will be extremely sensitive to the error. In particular, we will frequently conclude, incorrectly, that $Pr(B|\texttt{image}) \approx 1$ and $Pr(A|\texttt{image}) \approx 0$. This phenomenon, which is quite common in computer vision, makes it quite difficult to assess confidence values in recognition problems.

Rather than estimating posterior probabilities very accurately in order to be sure of certain results, we suggest an alternative. We formulate our confidence estimate as an hypothesis test that a certain result is *incorrect*, and if there is sufficient evidence, we reject the hypothesis that the result is incorrect. As we shall see, this comes closer to *bounding* the probabilities of certain results, which can be done with greater confidence, than *estimating* the probability of results, which is much more difficult. A critical aspect of our approach is that if there is insufficient evidence to reject a hypothesis, then we make no judgment on the correctness of the result. Our process only makes decisions when there is enough evidence, and avoids making decisions when there is not.

One interesting aspect of our work is that we make use of our bounding result as an important intermediate step in our overall system. In general, bounds given in machine learning are used to give theoretical justification for pursuing a particular algorithm and to gain insights on why they work. For instance, variational mean field inference can be viewed as optimizing a lower bound on the log partition function (Koller and Friedman, 2009).

In contrast, we make active use of our bound to guarantee that our document-specific training data will be nearly error-free. In this way, our bound plays in an integral role in the system itself, rather than as an analysis of the system.

2. Such errors are extremely difficult to avoid in high-dimensional estimation problems, since there is simply not enough data to estimate the exponents accurately.

2.1 OCR and Document-Specific Modeling

Despite claims to the contrary, getting OCR systems to obtain very high accuracy rates on moderately degraded documents continues to be a challenging problem (Nagy, 2000). One promising approach to achieving very high OCR accuracy rates is to incorporate *document-specific modeling* (Ho, 1998; Hong and Hull, 1995b,c). This set of approaches attempts to refine OCR models to specifically model the document currently being processed by adapting to the fonts in the document, adapting to the noise model in the document, or adapting to the lexicon in the document.

If one had some method for finding a sample of words in a document that were known to be correct with high confidence, one could effectively use the characters in such words as training data with which to build document-specific models of the fonts in a document. Resolving this circular-dependency problem is not easy, however.

To tackle this problem of producing "clean word lists" for document-specific modeling, we consider a somewhat different approach. Rather than trying to estimate the probability that an intermediate output of an OCR system (like an HMM or CRF) is correct and then thresholding this probability, we instead form a set of hypotheses about each word in the document. Each hypothesis poses that one particular word of the first-pass OCR system is incorrect. We then search for hypotheses that we can reject with high confidence. More formally, we treat a third party OCR system (in this case, the open source OCR program Tesseract (`http://code.google.com/p/tesseract-ocr/`) as a null hypothesis generator, in which each attempted transcription T produced by the OCR system is treated as the basis for a separate null hypothesis. The null hypothesis for word T is simply "*Transcription T is incorrect.*". Letting W be the true identity of a transcription T, we notate this as

$$T \neq W.$$

Our goal is to find as many hypotheses as possible that can be rejected *with high confidence*. In this paper, we take high confidence to mean with fewer than 1 error in a thousand rejected hypotheses. As we mention later, we only make 2 errors in 4465 words in our clean word lists, even when they come from quite challenging documents.

Before proceeding, we stress that the following are **not** goals of this paper:

- to present a complete end-to-end system for OCR,
- to produce accurate estimates of the probability of error of particular words in OCR.

Once again, our goal is to produce large lists of clean words from OCR output and demonstrate how they can be used for document-specific modeling. After presenting our method for producing clean word lists, we provide a formal analysis of the bounds on the probability of incorrectly including a word in our clean word list, under certain assumptions. When our assumptions hold, our error bound is very loose, meaning our true probability of error is much lower. However, some documents do in fact violate our assumptions.

We analyze this approach, and find that, with modest assumptions, we can bound the probability that our method produces an error at less than 0.002. Moreover, as a first-step validation of our general approach, we give a simple method for building a model from the document-specific data that significantly reduces the character error on a difficult, real-world data set.

We also compare our method with using the built-in confidence measure of a public domain OCR system, and thresholding this value to produce document-specific training data. We find that this method produces results that are less consistent and worse at reducing character error than our method.

2.2 Related Work

Our approach has ties with both prior work in OCR as well as methods outside of OCR, such as in image retrieval. We give a survey of related work below.

2.2.1 IN OCR

There has been significant work done in making use of the output of OCR in an iterative fashion, although all different from the work we present here. Kukich (1992) surveyed various methods to correct words, either in isolation or with context, using natural language processing techniques. Isolated-word error correction methods analyze spelling error patterns, for example, by deriving heuristics for common errors or by examining phonetic errors, and attempting to fix these errors through techniques such as minimum edit distance, *n*-gram statistics, and neural networks. Context-dependent word correction methods include using statistical language models such as word *n*-gram probabilities to correct errors using neighboring words.

Kolak (2003) developed a generative model to estimate the true word sequence from noisy OCR output. They assume a generative process that produces words, characters, and word boundaries, in order to model segmentation and character recognition errors of an OCR system. The model can be trained on OCR output paired with ground truth and then used to post-process and correct additional OCR output by finding the set of words, characters, and word boundaries that maximize the probability of the observed labeling.

Our work is distinguished from the above mentioned methods in that we examine the document images themselves to build document-specific models of the characters. A similar idea was used by Hong and Hull (1995a), who examined the inter-word relationships of character patches to help constrain possible interpretations. Specifically, they cluster whole word images and use majority voting of the associated OCR labels to decide on the correct output and create character image prototypes. This information is then used to correct additional errors by examining sub-patterns (e.g., a word is a prefix of another word) and decompositions of unknown words into known word patterns using the document images. Our work extends these ideas to produce clean, document-specific training data that can then be used in other methods, rather than only using potentially noisy labels through sub-pattern and decomposition analysis.

Our work is also related to a variety of approaches that leverage inter-character similarity in documents in order to reduce the dependence upon a priori character models. One method for making use of such information is to treat OCR as a cryptogram decoding problem, which dates back to Casey (1986) and Nagy (1986). After performing character clustering, decoding can be performed by a lexicon-based method (Ho and Nagy, 2000) or using hidden Markov models (Lee, 2002); however, such methods are limited by the assumption that characters can be clustered cleanly into pure clusters consisting of only one character. This particular problem can be overcome by solving the decoding problem iteratively, using word and character statistics to first decode least ambiguous characters, then to iteratively decode progressively more difficult characters (Kae and Learned-Miller, 2009).

An alternative approach to obtaining document-specific character models is presented by Edwards and Forsyth (2005), using an iterative algorithm to extract character templates from high confidence regions. One major difference is that we provide a theoretical bound on the number of errors expected using our algorithm to identify highly confident words. Another significant differ-

ence is that the authors provide a small amount of manually defined training data in their application, whereas we provide none.

Another method for leveraging inter-character similarity is to perform some type of character clustering. Hobby and Ho (1997) perform clustering in order to replace individual, potentially degraded character images, with a smoothed image over the cluster. Breuel (2003) learns a probabilistic similarity function to perform nearest-neighbor classification of characters.

The inability to attain high confidence in either the identity or equivalence of characters in these papers has hindered their use in subsequent OCR developments. We hope that the high confidence values we obtain will spur the use of these techniques for document-specific modeling.

2.2.2 OTHER WORK

Outside of OCR, our work is similar to Leisink and Kappen (2003), which deals with inference in graphical models for which exact inference is intractable. As an alternative to approximate inference techniques (which may bound a different quantity, the log partition function), they directly bound the marginal probabilities at each node in an iterative process called bound propagation. Each iteration consists of solving a linear program, where some of the constraints are due to bounds computed by previous iterations.

The end product of bound propagation is an upper and lower bound for each of the marginal probabilities of the nodes in the graphical model, with no guarantee on the tightness of any particular bound. In contrast, our work focuses on finding the subset of words for which we can put a very tight bound on the probability of error, and thus is a different approach under the general idea of bounding probabilities.

Our work is also related to the problem of covariate shift (Shimodaira, 2000), in which it is assumed that the conditional distributions $p(y|x)$ remain the same for both the training and test distributions, but the distribution on the observations $p(x)$ may differ. In this case, letting $p_0(x)$ be the distribution for the training set, and $p_1(x)$ be the distribution for a test set, one can reweight the log likelihood of the training instances with $\frac{p_1(x)}{p_0(x)}$. The principal difficulty is estimating this ratio. In particular, in OCR, test documents may have a range of degradation and noise, and potentially unseen font models, and thus the support of $p_0(x)$ may potentially not contain the support of $p_1(x)$, in which case a re-weighting approach could not be applied. Moreover, noise and font appearance specific to the test document may also lead to a change in $p(y|x)$ for ambiguous or noisy x. Instead, our work attempts to identify highly confident labelings (x',y') in order to characterize the test-specific distribution over appearance and labels.

Another area closely related to the method presented in this paper is the meta-recognition work of Scheirer et al. (2011). They consider the problem of multiclass recognition, such as object or face recognition. A given test image produces a set of scores indicating how well the test image matched each class. Since the test image can belong to at most one class, all but the highest returned score can be used to model the distribution of non-matching scores, specific to the single test image. The authors use some fraction of the top non-matching scores produced for a test image to model the tail of the non-matching distribution using extreme value theory, and then use this distribution to normalize the top matching score.

Similar to our work, the tail distribution that is modeled can be used to attempt to reject the null hypothesis that the top matching score belongs to the non-matching distribution. Our work differs in that we specifically focus only on cases where we can reject this null hypothesis with very high

confidence. To do so, we leverage the appearance of the entire document, which allows us to be more robust to cases where the test distribution differs substantially from the training distribution.

The idea of identifying objects which can confidently be given a particular label is also an important component of query expansion in the information retrieval field. Query expansion is a technique used to add terms to an initial query based on the highly ranking documents of the initial query. In Chum et al. (2007), query expansion is used for image retrieval where the initial results of an image query are processed to find resulting images that the system is confident match the initial query. The confidence in a particular match is evaluated using a spatial verification scheme that is similar to our consistency check presented below. This verification is critical to query expansion, as false positives can lead to drift, causing irrelevant features to be added to the expanded query. Later, we propose a possible extension to see whether our bound analysis can be applied to give a bound on the probability of a false match passing the spatial verification.

Building models specific to a test image has also been applied in other areas of computer vision. In work by Nilsback and Zisserman (2007), an initial, general flower model is applied to an image to segment a flower from the background. This initial segmentation is used to build an image-specific color model of the foreground flower, and this process of segmentation and color estimation is iterated until convergence.

Berg et al. (2007) follow a similar approach to image parsing, first extracting a per pixel segmentation of the image, then using pixels with high confidence to learn an image-specific color model of sky and building. Ramanan (2006) uses an initial edge model to infer the pose of a person in the image, then uses this to build an image-specific color model, and iterates until convergence. These methods can be sensitive to the initial steps, underscoring the need for high precision in constructing image-specific models. Sapp et al. (2010) take a slightly different approach by using similarity between a test image and a set of training exemplars and kernel regression to learn image-specific model parameters, and then performing inference with the image-specific model.

3. Method for Producing Clean Word Lists

In this section, we present our method for examining a document bitmap and the output of an OCR system for that document to produce a so-called *clean word list*, that is, a list of words which we believe to be correct, with high confidence. Our success will be measured by the number of words that can be produced, and whether we achieve a very low error rate in the clean list. Ideally, we must produce a clean word list which is large enough to provide sufficient training data for document-specific modeling.

We assume the following setup.

- We are provided with a document D in the form of a grayscale image.
- We are provided with an OCR system.
- We further assume that the OCR system provides an *attempted* segmentation of the document D into words, and that the words are segmented into characters. It is not necessary that the segmentation be entirely correct, but merely that the system produces an attempted segmentation.
- In addition to a segmentation of words and letters, the system should produce a best guess for every character it has segmented, and hence, by extension, of every word (or string) it has

segmented. Of course, we do not expect all of the characters or words to be correct, as that would make our exercise pointless.

- Using the segmentations provided by the OCR system, we assume we can extract the gray-valued bitmaps representing each guessed character from the original document image.

- Finally, we assume we are given a lexicon. Our method is relatively robust to the choice of lexicon, and assumes there will be a significant number of non-lexicon words in the document.

We define a few terms before proceeding. The *Hamming distance* between two strings of the same number of characters is the number of character substitutions necessary to convert one string to the other. The *Hamming ball* of radius r for a word W, $H_r(W)$, is the set of strings whose Hamming distance to W is less than or equal to r. Later, after defining certain equivalence relationships among highly confusable characters such as '`o`' and '`c`', we define a *pseudo-Hamming distance* which is equivalent to the Hamming distance except that it ignores substitutions among characters in the same equivalence class. We also use the notions of edit distance, which extends Hamming distance by including joins and splits of characters, and pseudo-edit distance, which is edit distance using the aforementioned equivalence classes.

Our method for identifying words in the clean list has three basic steps. We consider each word T output by the initial OCR system.

1. If T is not in the lexicon, we discard it and make no attempt to classify whether it is correct. That is, we do not put it on the clean word list.[3]

2. Given that T is a lexicon word, we evaluate whether $H_1(T)$ is non-empty, that is, whether there are any lexicon words for which a single change of a letter can produce T. If $H_1(T)$ is non-empty, we discard T and again make no attempt to classify whether it is correct.

3. Assuming we have passed the first two tests, we now perform a *consistency check* (described below) of each character in the word. If the consistency check is passed, we declare the word to be correctly recognized and include it in the clean list.

3. Why is it not trivial to simply declare any output of an OCR system that is a lexicon word to be highly confident? The reason is that OCR systems frequently use language models to project uncertain words onto nearby lexicon words. For example, suppose the original string was "Rumpledpigskin", and the OCR system, confused by its initial interpretation, projected "Rumpledpigskin" onto the nearest lexicon word "Rumplestiltskin". A declaration that this word is correct would then be wrong. However, our method will not fail in this way because if the true string were in fact "Rumpledpigskin", the character consistency check would never pass. It is for this reason that our method is highly non-trivial, and represents a significant advance in the creation of highly accurate clean word lists.

   We could potentially restrict our attention to OCR systems that did not project onto lexicon words, or for which it is possible to access intermediate results prior to such projection. For such results, it is much more likely that a word labeled as a lexicon word with an empty Hamming ball of some radius is, in fact, correctly labeled. We choose not to make such a restriction, both so that our method is more general, and because projecting uncertain words to nearby lexicon words can often substantially increase the labeling accuracy. In other words, by only considering labelings obtained without such projection, we may find far fewer words that we can confidently classify as being correctly labeled, due to the lower accuracy of the initial OCR system. The benefit of using a lexicon is evident in the scene text recognition work of Weinman et al. (2009). In this work, simply forcing all predicted words to be lexicon words led to a 3 percentage point increase in word accuracy, and incorporating factors with lexicon information into the probability model led to an additional 5 percentage point increase in word accuracy. By performing a more robust analysis than accepting lexicon words, our method is equally applicable to sophisticated OCR systems that make use of lexicon information.

3.1 Consistency Check

In the following discussion, we use the term *glyph* to refer to a rectangular portion of an image that is likely to be a single character, but may be only a portion of a character, multiple characters, or a stray mark. Let W_j be the true character class of the jth glyph of a word W, and let T_j be the initial OCR system's interpretation of the same glyph. The goal of a consistency check is to ensure that the OCR system's interpretation of a glyph is reliable. We will assess reliability by checking whether other similar-looking glyphs are usually interpreted the same by the OCR system.

To understand the purpose of the consistency check, consider the following situation. Imagine that a document contains a stray mark that does not look like any character at all, but was interpreted by the initial OCR system as a character. If the OCR system thought that the stray mark was a character, it would have to assign it to a character class like 't'. We would like to detect that this character is unreliable. Our scheme for doing this is to find other characters that are similar to this glyph, and to check the identity assigned to those characters by the initial OCR system. If a large majority of those characters are given the same interpretation by the OCR system, then we consider the original character to be reliable. Since it is unlikely that the characters closest to the stray mark are clustered tightly around the true character 't', we hope to detect that the stray mark is atypical, and hence unreliable.

More formally, to test a glyph g for reliability, we first find the M glyphs in the document that are most similar to g (using normalized correlation as the similarity measure). If a fraction θ of the M glyphs most similar to g have the character label c, then we say that the glyph g is θ-dominated by c. More precisely, we run the following procedure:

```
// n : vector storing the counts for each character c.
// L : set of character labels.
// M : number of glyphs to compare to.
n[c] ← 0, ∀c ∈ L
for i ← 1 to M do
    c ← label of character ith most similar to g
    n[c] = n[c] + 1
    if n[c]/(i+1) > θ then
        return g is θ-dominated by c
    end
end
return g is undominated
```

Algorithm 1: Consistency check algorithm.

There are three possible outcomes of the consistency check. The first is that the glyph g is dominated by the same class c as the OCR system's interpretation of g, namely T_j. The second outcome is that g is dominated by some other class that does not match T_j. The third outcome is that g is undominated, meaning that the neighbors of g are relatively inconsistent. In the latter two cases, we declare the glyph g to be *unreliable*. The interpretation of glyph g is reliable only if g is dominated by the same class as the original OCR system. Furthermore, a word is included in the clean list only if all of the characters in the word are reliable.

The constants used in our experiments were $M = 20$ and $\theta = 0.66$. That is, we compared each glyph against a maximum of 20 other glyphs in our reliability check, and we insisted that a "smoothed" estimate of the number of similarly interpreted glyphs was at least 0.66 before declaring

a character to be reliable. We now analyze the probability of making an error in the clean set, under a specific set of assumptions.

4. Theoretical Bound

For a word in a document, let W be the ground truth label of the word and T be the initial OCR system's labeling of the word. Consider the problem of trying to estimate the probability that the labeling was correct, $P(W = w_t|T = w_t)$. It is difficult to formulate a bound or performance guarantee on such an estimate, due to the non-stationarity in the sequence of words. The distribution and appearance of words is dependent on the topics and fonts present in the document containing the words, and any noise in the document, which may range from local, such as stray marks, to global, such as low contrast. Therefore, we would not be able to rely on a general *i.i.d.* assumption on the words.

Rather than attempting to estimate the probability that the labeling was correct, we circumvent the above problems by focusing on bounding the probability for a subset of words. Let C be a binary indicator equal to 1 if the word passed the consistency check. We want to upper bound the probability $\Pr(W \neq w_t|T = w_t, C = 1)$ when w_t is a lexicon word and has an empty Hamming ball of size 1. We decompose the probability into three terms:

$$\begin{aligned}
\Pr(W \neq w_t|T = w_t, C = 1) &= \sum_{w \neq w_t} \Pr(W = w|T = w_t, C = 1) \\
&= \sum_{w \neq w_t, w \in \text{Lex}} \Pr(W = w|T = w_t, C = 1) \\
&\quad + \sum_{w \neq w_t, w \notin \text{Lex}} \Pr(W = w|T = w_t, C = 1) \\
&= \sum_{w \neq w_t, w \in \text{Lex}, |w| = |w_t|} \Pr(W = w|T = w_t, C = 1) \qquad (1) \\
&\quad + \sum_{w \neq w_t, w \in \text{Lex}, |w| \neq |w_t|} \Pr(W = w|T = w_t, C = 1) \\
&\quad + \sum_{w \neq w_t, w \notin \text{Lex}} \Pr(W = w|T = w_t, C = 1).
\end{aligned}$$

Our approach for bounding this probability will be to individually bound the three terms in Equation 1. The first term considers all words in the lexicon with the same length as w_t, and accounts for the most likely type of error. The second term considers all words in the lexicon, but with a different length from w_t, and so considers many more possible words resulting from segmentation errors, but each of which is much less likely to occur. Finally the third term considers words not in the lexicon, each of which occurs even less frequently.

To bound the first term, we can consider all words of a given Hamming distance i from w_t. Our strategy will then be to bound the contribution to the error from all words of Hamming distance i, enabling us to bound the total error as the sum of a geometric series. We can then follow the same approach to bound the next two terms, where we will instead need to consider edit distance rather than Hamming distance.

In order to bound these terms using a geometric series, we will need two initial steps. We will need an upper bound on the probability of the consistency check passing for a specific character when the label is incorrect (2ε), and a lower bound on the probability of the consistency check

passing when the label is correct (δ). We explain these quantities in the next section. Next, we will need to relate these bounds on the character consistency check, $\Pr(C = 1|T = w_t, W)$, to the terms in the geometric series, $\Pr(W = w|T = w_t, C = 1)$, which we do in Section 4.2.

4.1 Bounding the Character Consistency Check

We will rewrite the $\Pr(W = w|T = w_t, C = 1)$ terms as bounds involving $\Pr(C = 1|T = w_t, W = w)$ using Bayes' rule. We will make the assumption that the individual character consistency checks are independent, although this is not exactly true, since there may be local noise that degrades characters in a word in the same way.

Assume that each character is formed on the page by taking a single true, latent appearance based on the font and the particular character class and adding some amount of noise. Let ε be an upper bound on the probability that noise has caused a character of any given class to look like it belongs to another specific class other than its own class. More formally, letting $p_c(a)$ be the probability of a character appearance a for a given class c under the noise model, ε satisfies, for all character classes $c_1, c_2, c_1 \neq c_2$,

$$\varepsilon > \int_{a|p_{c_1}(a)<p_{c_2}(a)} p_{c_1}(a)da. \tag{2}$$

In order to obtain a small value for ε, and hence later a small probability of error, we revise Equation 2 to be a bound only on *non-confusable* character classes. In other words, since some character classes are highly confusable, such as 'o', 'c', and 'e', we ignore such substitutions when computing Hamming and edit distance. We'll refer to these distances as pseudo distances, so "mode" and "mere" have a true Hamming distance of 2 but a pseudo-Hamming distance of 1.

This is similar to defining an equivalence relation where confusable characters belong to the same equivalence class, and computing distance over the quotient set, but without transitivity, as, for example, 'h' may be confusable with 'n', and 'n' may be confusable with 'u', but 'h' may not necessarily be confusable with 'u'.

For a character to pass a consistency check with the label c_2 when the true underlying label is c_1, roughly one of two things must happen: (a) either the character was corrupted and looked more like c_2 than c_1, or (b) some number of other characters with label c_2 were corrupted and looked like c_1's.

The probability (a) is clearly upper bounded by ε, since it requires both the corruption and most of its neighbors to have the same label c_2. Since $\varepsilon \ll 1$ and (b) requires several other characters with label c_2 to be corrupted to look like c_1, the probability of (b) should be bounded by (a), and thus ε, as well. Therefore the probability of the consistency check giving a label c_2 when the true underlying label is c_1 is less than 2ε, for any classes c_1, c_2.

We will also need a lower bound on the probability that a character consistency check will succeed if the OCR system's label of the character matches the ground truth label. Let δ be a lower bound on this quantity, which is dependent on both the amount of noise in the document and the length of the document. (The latter condition is due to the fact that the character consistency check requires a character to match to at least a certain number of other similarly labeled characters, so, for example, if that number is not present in the document to begin with, then the check will fail with certainty.)

4.2 Bounding One Term

Consider bounding $\Pr(W = w|T = w_t, C = 1)$:

$$\begin{aligned}
&\Pr(W = w|T = w_t, C = 1) \\
&\quad = \frac{\Pr(C = 1|T = w_t, W = w)\Pr(W = w|T = w_t)}{\sum_{w'}\Pr(C = 1|T = w_t, W = w')\Pr(W = w'|T = w_t)} \\
&\quad = \frac{\Pr(C = 1|T = w_t, W = w)\Pr(T = w_t|W = w)\Pr(W = w)}{\sum_{w'}\Pr(C = 1|T = w_t, W = w')\Pr(T = w_t|W = w')\Pr(W = w')} \\
&\quad \leq \frac{\Pr(C = 1|T = w_t, W = w)\Pr(T = w_t|W = w)\Pr(W = w)}{\Pr(C = 1|T = w_t, W = w_t)\Pr(T = w_t|W = w_t)\Pr(W = w_t)}. \qquad (3)
\end{aligned}$$

Here the inequality follows from the fact that w_t is one of the words being summed over in the denominator, and hence replacing the sum with only the w_t component will make the denominator less than or equal to the sum.

4.3 Bounding the Probability of Lexicon Words

Recall that w_t, the initial OCR system's word labeling, is a lexicon word with empty pseudo-Hamming ball of size 1. For lexicon words w, we will assume that

$$\frac{\Pr(T = w_t|W = w)\Pr(W = w)}{\Pr(T = w_t|W = w_t)\Pr(W = w_t)} \quad < \quad 1,$$

or, equivalently,

$$\frac{\Pr(W = w|T = w_t)}{\Pr(W = w_t|T = w_t)} \quad < \quad 1. \qquad (4)$$

One way to view this is to think of $T = w_t$ as a feature. Then, for a reasonable classifier, this assumption should hold for any document in the training set, as this is simply the Bayes decision rule. (If the assumption did not hold, then we could increase the training accuracy by predicting w whenever we saw the feature $T = w_t$.)

Thus, we are assuming that a test document does not differ from the training documents used to train the initial OCR system so much as to change the most probable word conditioned on the feature $T = w_t$, as suggested by Equation 4.[4] Note that w_t has an empty Hamming ball of size 1, so w differs from w_t by at least two letters. For this assumption to be violated, either the document must be such that at least one letter is consistently interpreted as another, or has an extremely different prior distribution on words than that of the training set, both of which are unlikely. As we discuss later, the first case is also problematic for the character consistency check as well, and so falls outside the scope of documents for which our method will be applicable.

It is important to note that this does not imply that the word accuracy need be particularly high, for example, if all the words have the same prior probability of occurring, then the assumption could hold for a classifier with accuracy simply better than the chance accuracy of $\frac{1}{|\text{Lex}|}$, where $|\text{Lex}|$ is the size of the lexicon.

Applying this to Equation 3, we get

$$\Pr(W = w|T = w_t, C = 1) \quad \leq \quad \frac{\Pr(C = 1|T = w_t, W = w)}{\Pr(C = 1|T = w_t, W = w_t)}. \qquad (5)$$

4. This assumption is similar to, but slightly weaker than, the assumption made under covariate shift (Shimodaira, 2000).

4.3.1 BOUNDING THE PROBABILITY OF LEXICON HAMMING WORDS

Consider a lexicon word w that is a pseudo-Hamming distance i from w_t. We can then simplify Equation 5 to

$$\Pr(W = w|T = w_t, C = 1) \quad \leq \quad \frac{(2\varepsilon)^i}{\delta^i}$$

by making use of the assumption that the character consistency checks are independent, and that w and w_t only differ in i characters. For those i characters, w does not match the OCR system's label and w_t does match the OCR system's label, so we use the bounds 2ε and δ.

Now let D_i be the number of lexicon words of pseudo-Hamming distance i away from w_t. Let r_D be the rate of growth of D_i as a function of i, that is, $D_{i+2} \leq r_D^i D_2$. Assume, since $\varepsilon \ll 1$, that $r_D(\frac{2\varepsilon}{\delta}) < \frac{1}{2}$.[5] (To produce a final number for our theoretical bound, we later assume that $\varepsilon < 10^{-3}$ and $\delta^2 > 10^{-1}$. Given these numbers, our assumption becomes $r_d < 79$, which is a very conservative bound as experiments on the lexicon used in our main experiments showed that r_D is generally bounded by 5.)

To get the total contribution to the error from all lexicon Hamming words, we sum over D_i for all $i > 1$,

$$\begin{aligned}
\sum_{w \neq w_t, w \in \text{Lex}, |w| = |w_t|} \Pr(W = w|T = w_t, C = 1) \quad &\leq \quad \sum_{i=2} D_i \frac{(2\varepsilon)^i}{\delta^i} \\
&= \quad D_2 \frac{(2\varepsilon)^2}{\delta^2} + D_2 \frac{(2\varepsilon)^2}{\delta^2} \sum_{i=1} (2 r_D \frac{\varepsilon}{\delta})^i \\
&\leq \quad 8 D_2 \frac{\varepsilon^2}{\delta^2}.
\end{aligned}$$

4.3.2 BOUNDING LEXICON EDIT WORDS

Traditionally, edit distance is computed in terms of number of substitutions, insertions, and deletions necessary to convert one string to another string. In our context, a more natural notion may be splits and joins rather than insertions and deletions. For example, the interpretation of an '`m`' may be split into an '`r`' and an '`n`', or vice-versa for a join.

The probability that a split or a join passes the consistency check is upper bounded by $(2\varepsilon)^2$. We can see this from two perspectives. First, a split or join has traditional edit distance of 2, since it requires an insertion or deletion and a substitution ("`m`" to "`mn`" insertion followed by "`mn`" to "`rn`" substitution).

A more intuitive explanation is that, for a split, one character must be corrupted to look like the left hand side of the resulting character and another character corrupted to look like the right hand side, and for a join, the left hand side of a character must be corrupted to look like one character and the right hand side corrupted to look like another.

Similar to the case of confusable characters for substitutions, we also ignore confusable characters for splits and joins, namely '`r`' followed by '`n`' with '`m`', and '`v`' followed by '`v`' with '`w`'. Thus, "`corn`" and "`comb`" have an edit distance of 2 but a pseudo-edit distance of 1.

5. Recall that δ, as defined earlier, is a lower bound on the probability that a character consistency check will succeed if the OCR system's label of the character is correct.

Consider a lexicon word w with pseudo-edit distance i from w_t, and involving at least one insertion or deletion (so $|w| \neq |w_t|$). Similar to the lexicon Hamming words, we can simplify Equation 5 for w as

$$\Pr(W = w|T = w_t, C = 1) \quad \leq \quad \frac{(2\varepsilon)^{i+1}}{\delta^i},$$

since each substitution contributes a $\frac{2\varepsilon}{\delta}$ and each insertion or deletion, of which there is at least one, contributes a $\frac{(2\varepsilon)^2}{\delta}$.

Let E_i be the number of lexicon words w with a pseudo-edit distance i away from w_t and $|w| \neq |w_t|$. Again, also assume that r_E, the rate of growth of E_i, satisfies $r_E(\frac{2\varepsilon}{\delta}) < \frac{1}{2}$. Summing the total contribution to the error from lexicon edit words,

$$\begin{aligned}
\sum_{w \neq w_t, w \in \text{Lex}, |w| \neq |w_t|} \Pr(W = w|T = w_t, C = 1) &\leq \sum_{i=1} E_i \frac{(2\varepsilon)^{i+1}}{\delta^i} \\
&= E_1 \frac{(2\varepsilon)^2}{\delta} + E_1 \frac{(2\varepsilon)^2}{\delta} \sum_{i=1} (2r_E \frac{\varepsilon}{\delta})^i \\
&\leq 8E_1 \frac{\varepsilon^2}{\delta} \\
&\leq 8E_1 \frac{\varepsilon^2}{\delta^2}.
\end{aligned}$$

4.4 Bounding Non-Lexicon Words

Let N_i be the set of non-lexicon words with a pseudo-edit distance i from w_t, and let $p_i = \frac{\Pr(T=w_t|W \in N_i)\Pr(W \in N_i)}{\Pr(T=w_t|W=w_t)\Pr(W=w_t)}$. Assume the rate of growth of r_N of p_i satisfies $r_N(\frac{2\varepsilon}{\delta}) < \frac{1}{2}$.

Rearranging Equation 3 and summing over all non-lexicon words:

$$\begin{aligned}
&\sum_{w \neq w_t, w \notin \text{Lex}} \Pr(W = w|T = w_t, C = 1) \\
&\quad \leq \sum_{i=1} \sum_{w \in N_i} \frac{\Pr(C = 1|T = w_t, W = w)\Pr(W = w|T = w_t)}{\Pr(C = 1|T = w_t, W = w_t)\Pr(W = w_t|T = w_t)} \\
&\quad \leq \sum_{i=1} \sum_{w \in N_i} \frac{(2\varepsilon)^i}{\delta^i} \frac{\Pr(W = w|T = w_t)}{\Pr(W = w_t|T = w_t)} \\
&\quad = \sum_{i=1} \frac{(2\varepsilon)^i}{\delta^i} \frac{\Pr(W \in N_i|T = w_t)}{\Pr(W = w_t|T = w_t)} \\
&\quad = \sum_{i=1} \frac{(2\varepsilon)^i}{\delta^i} p_i \\
&\quad \leq p_1 \frac{2\varepsilon}{\delta} + p_1 \frac{2\varepsilon}{\delta} \sum_{i=1} (2r_N \frac{\varepsilon}{\delta})^i \\
&\quad \leq 4p_1 \frac{\varepsilon}{\delta^2}.
\end{aligned}$$

4.5 Final Bound

Combining each of the individual bounds derived above, we have

$$\Pr(W \neq w_t | T = w_t, C = 1) \quad \leq \quad \frac{(8D_2 + 8E_1)\varepsilon^2 + 4p_1\varepsilon}{\delta^2}.$$

To use this in practice, we need to set some realistic (but conservative) values for the remaining constants. For $\varepsilon < 10^{-3}$, $8D_2 + 8E_1 < 10^2$, $4p_1 < 10^{-1}$, $\delta^2 > 10^{-1}$,

$$\Pr(W \neq w_t | T = w_t, C = 1) \quad \leq \quad 2 \cdot 10^{-3}.$$

The bounds for the constants chosen above were selected conservatively to hold for a large range of documents, from very clean to moderately noisy. Not all documents will necessarily satisfy these bounds. In a sense, these inequalities define the set of documents for which our algorithm is expected to work, and for heavily degraded documents that fall outside this set, the character consistency checks may no longer be robust enough to guarantee a very low probability of error.

Our final bound on the probability of error, 0.002, is the result of a *worst case analysis* under our assumptions. If our assumptions hold, the probability of error will likely be much lower for the following reasons. For most pairs of letters, $\varepsilon = 10^{-3}$ is not a tight upper bound. The quantity on the right of Equation 4 is typically much lower than 1. The rate of growths r_D, r_E, r_N are typically much lower than assumed. The bound on p_1, the non-lexicon word probabilities, is not a tight upper bound, as non-lexicon words mislabeled as lexicon words are rare. Finally, the number of Hamming and edit distance neighbors D_2 and E_1 will typically be less than assumed.

On the other hand, for sufficiently noisy documents, and certain types of errors, our assumptions do not hold. Some of the problematic cases include the following. As discussed, the assumption that the individual character consistency checks are independent is not true. If a document is degraded or has a font such that one letter is consistently interpreted as another,[6] then that error will likely pass the consistency check (i.e., ε will be very large). If a document is degraded or is very short, then δ may be much smaller than $10^{-\frac{1}{2}}$. (The character consistency check requires a character to match to at least a certain number of other similarly labeled characters, so, for example, if that number isn't present in the document to begin with, then the check will fail with certainty.) Finally, if the lexicon is not appropriate for the document then $4p_1 < 10^{-1}$ may not hold. This problem is compounded if the OCR system projects to lexicon words. Still these assumptions appear to hold for a wide range of documents.

5. Character Recognition

To validate the utility of our clean word lists, we implemented a simple technique for constructing document-specific character appearance models, using SIFT features (Lowe, 2004),[7] and demonstrated that this model can be used to significantly reduce character error in the remainder of the document. We refer to our algorithm as SIFT_Align. In the future, we believe these clean word lists can be incorporated into more sophisticated document-specific OCR models to obtain further improvements in recognition accuracy, as we discuss in future work.

6. It should be noted that the probability of such an error (consistently interpreting one letter as another) is substantially reduced by using a language model, for example, projecting uncertain words to nearby lexicon words.
7. A SIFT feature is essentially computed by dividing the image into a set of non-overlapping patches, and computing a histogram over edge orientations weighted by edge strength, for each patch.

We use the traditional SIFT descriptor without applying the Gaussian weighting because we did not want to weight the center of an image more highly than the rest. In addition, we fix the scale to be 1 and orientation to be 0 at all times. The SIFT_Align procedure is presented below:

1. Compute the SIFT descriptor for each character image in the clean list, at the center of the image.

2. Compute the component-wise arithmetic mean of all SIFT descriptors for each character class in the clean list. These mean descriptors are the "representations" (or character models) of the respective classes.

3. For each character image in the clean list, compute a SIFT descriptor for each point in a window in the center of the image (we use a 5x5 window) and select the descriptor with smallest L2 distance to the mean SIFT descriptor for this character class. This aligns each character's descriptor to the mean class descriptor.

4. Test images are defined as follows. We start by collecting all character images that are *not* in the clean list (since we do not want to test on images we trained on). We also filter the test images as follows. If Tesseract gives a label to an image that is *not* a label for any of the clean set characters, then we do not include this character in our test set. The rationale for this is the following. Since our method will only assign to a character a label that appears in the clean set, then if the character was originally correct, we will definitely introduce an error by attempting to correct it. Furthermore, we have no direct information about the appearance of characters whose labels are *not* in the clean set, so it is relatively difficult to assess if the original label is unreasonable. For these reasons, we only attempt to correct characters whose Tesseract label appears as one of the clean set labels.

5. For each test image, again compute a 5x5 window of 25 SIFT descriptors, and select the descriptor which has minimum L2 distance to *any* of the mean descriptors. This aligned descriptor is the final descriptor for the test image.

6. Pass the SIFT descriptors for the training/test images found in the previous steps to a multi-class SVM.

In summary, this classifier can be described as simply using an SVM with SIFT descriptors, except that care is taken to align characters as well as possible for both training and testing. We use the $SVM^{multiclass}$ implementation[8] of multiclass SVM (Tsochantaridis et al., 2004) and use a high C value of 5,000,000, which was selected through cross-validation. This makes sense since we generally do not have many instances of each character class in the clean list, and so we want a minimum of slack, which a high C value enforces.

6. Experiments

In this section, we describe three types of experiments. First, we show that our procedure for generating clean sets achieves the very low error rate predicted by our bounds. Next, we show that for a collection of 56 documents, using the clean sets to train new, document-specific classifiers

8. SVM implementation can be found at `http://svmlight.joachims.org/`.

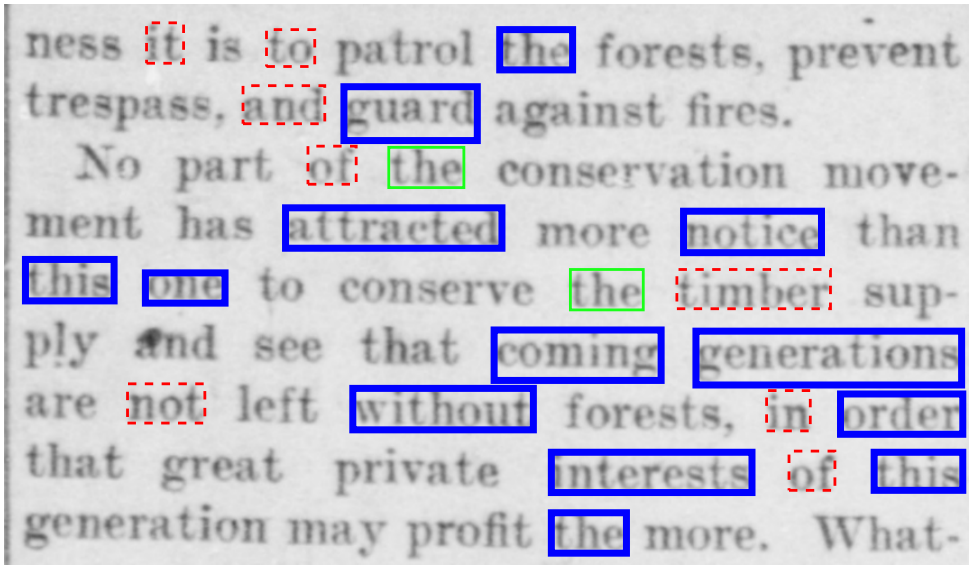

Figure 1: Thick blue boxes indicate clean list words. Dashed red boxes indicate Tesseract's confident word list. Thin green boxes indicate words in both lists. Despite being in Tesseract's list of high confidence words, "timber" is misrecognized by Tesseract as "timhcr". All other words in boxes were correctly translated by Tesseract. (Best viewed in color.)

significantly reduces OCR errors over the initial OCR system used. Finally, we show what happens if a traditional measure of confidence is used to select a document-specific training set. In particular, we show that our clean sets have far fewer errors and result in document-specific models that can correct a much larger number of errors in the original OCR output.

6.1 Initial Clean Set Experiments

We experimented with two sets of documents. The first set consists of 10 documents from the JSTOR archive[9] and Project Gutenberg.[10] This initial set of documents was used to evaluate our clean list generation algorithm and develop our algorithm for producing character models from the clean lists (Kae et al., 2009). In this work, our clean lists selected an average of 6% of the words from each document. *These clean lists did not contain a single error*, that is, the precision of our clean lists was 100%. This strongly supports our theoretical bounds established in Section 4.

9. JSTOR can be found at `http://www.jstor.org`.
10. Project Gutenberg can be found at `http://www.gutenberg.org/`.

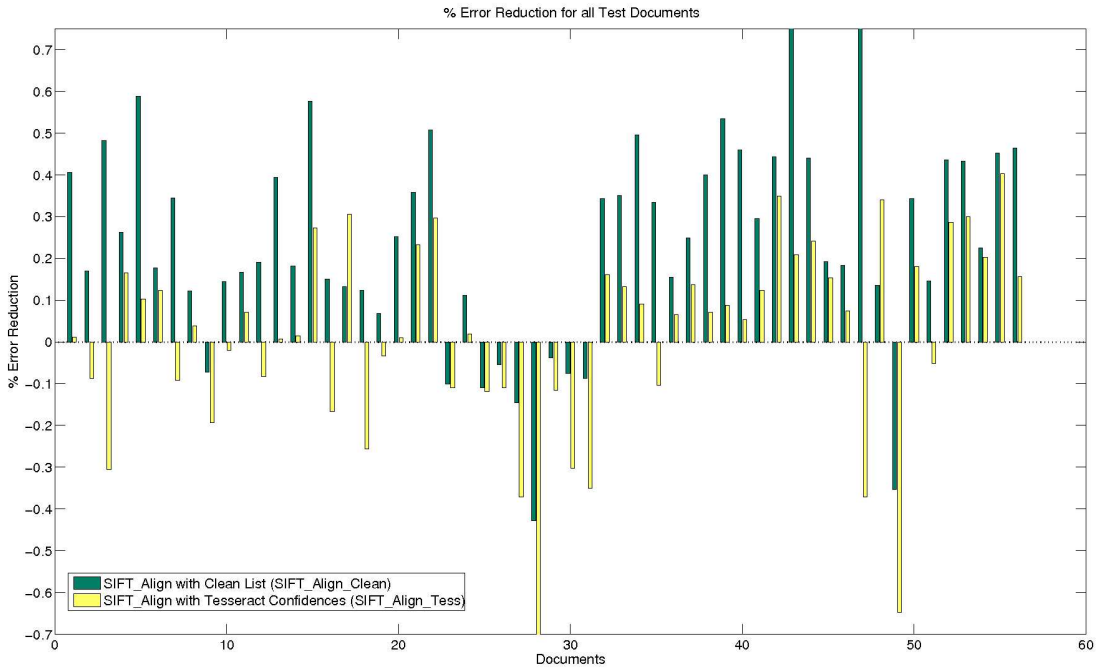

Figure 2: Character error reduction rates for SIFT_Align using the clean list (SIFT_Align_Clean) and Tesseract's confident word list (SIFT_Align_Tess) on the test sets of 56 documents. SIFT_Align_Clean increases the error rate in 10 documents whereas SIFT_Align_Tess increases the error rate in 21 documents.

6.2 Correcting OCR Errors

After establishing the basic viability of the clean set procedure, we selected another set of documents on which to test our end-to-end system of generating clean sets, using them to build document-specific models, and using these models, in turn, to correct errors made by the original OCR system.

The second set of documents, used for performance evaluation of the SIFT_Align algorithm, are 56 documents taken from the Chronicling America[11] archive of historical newspapers. Since our initial OCR system (Tesseract) can only accept blocks of text and does not perform layout analysis, we manually cropped out single columns of text from these newspaper pages. Other than cropping and converting to the TIFF image format for Tesseract, the documents were not modified in any way. There are on average 1204 words per document. The clean list contains 2 errors out of a total of 4465 words, within the theoretical bound of 0.002 mentioned earlier.

In an effort to increase the size of the clean lists beyond 6% per document, we experimented with relaxing some of the criteria used to select the clean lists. In particular, we allowed the Hamming ball of radius 1 for a word to be non-empty as long as the words within the ball did not appear within the original OCR system's translation. By making this small change, we were able to increase the size of the clean lists to an average of 18% per document while introducing at most one error per document. We refer to the original clean lists as *conservative clean lists* and to the modified, larger,

11. Documents can be found at `http://chroniclingamerica.loc.gov/`.

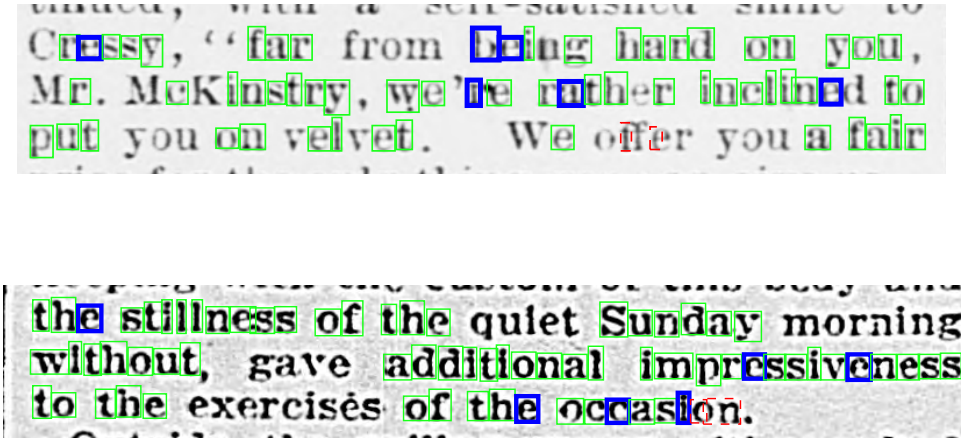

Figure 3: Sample of results from two documents. A thin green box indicates both the initial OCR system (Tesseract) and SIFT_Align correctly classified the character. A dashed red box indicates both systems misclassified the character, and a thick blue box indicates that SIFT_Align classified the character correctly and Tesseract misclassified it. In this example, there are no cases shown where Tesseract correctly classified a character and SIFT_Align misclassifies it. (Best viewed in color.)

and slightly less accurate clean lists as *aggressive clean lists*. We decided to use the aggressive clean lists for our experiments because they contain few errors and there are more character instances.[12] From this point, our use of "clean list" refers to the aggressive clean list.

We then ran Tesseract on all documents, obtaining character bounding boxes[13] and guesses for each character. Next, we used Mechanical Turk[14] to label all character bounding boxes to produce a ground truth labeling. We instructed annotators to only label bounding boxes for which a single character is clearly visible. Other cases (multiple characters in the bounding box or a partial character) were discarded.

After the initial OCR system was used to make an initial pass at each document, the clean list for that document was extracted. Character recognition was then performed as described in Section 5. Even though many of the characters were already recognized correctly by the original

12. To account for the looser criteria of the aggressive set, we would need to add a term to the theoretical bound that considers the probability of error due to the true labeling of the word being a neighbor of Hamming distance 1 from the OCR system's interpretation. This term would be $D_1 q \frac{2\varepsilon}{\delta}$, where D_1 is the number of neighbors of Hamming distance 1, and q is the probability that a word that was not detected anywhere in the document actually appears in the document. Given our assumed bounds of $\varepsilon < 10^{-3}$, $\delta^2 > 10^{-1}$, if we further assume $D_1 q$ to be conservatively bounded by 0.3, then using the aggressive criteria doubles the probability of error to 0.004.

    We note that the assumptions we make to produce the theoretical bound are very conservative. This leaves room for some experimentation to find the optimal balance between probability of error and clean set size, while still maintaining an empirical error close to the predicted bound of 0.002.
13. This feature is not available out of the box; we edited the source code.
14. Mechanical Turk can be found at `https://www.mturk.com/mturk/welcome`.

OCR system, our approach improves the recognition to produce an even higher accuracy than the original OCR system's accuracy, on average. As shown in the next section, in most cases, this resulted in correcting a significant portion of the characters in the documents.

6.3 Comparison to Another Confidence Measure

In order to judge the effectiveness of using our clean list, we also generated another confident word list using Tesseract's own measure of confidence.[15] To generate the confident word list, we sort Tesseract's recognized words by their measure of confidence and take the top *n* words that result in the same number of characters as our clean list.

In Figure 1, we show a portion of a document and the corresponding subset of clean list words (generated by our process) and highly confident Tesseract words. All of the words in our clean list were, in fact, correctly labeled by the initial Tesseract pass. In other words, our clean list for this example was error free. But Tesseract's high confidence word list includes "timber" which was mistranslated by Tesseract.

We refer to the SIFT_Align algorithm using our clean list as SIFT_Align_Clean and the SIFT_Align algorithm using Tesseract's confidences as SIFT_Align_Tess. In Figure 2, we show the character error reduction rates for both SIFT_Align_Clean and SIFT_Align_Tess. In 46 of the 56 documents, SIFT_Align_Clean results in a reduction of errors whereas SIFT_Align_Tess reduces error in 35 documents. Note this figure shows percent error reduction, not the raw number of errors. SIFT_Align_Clean made a total of 2487 character errors (44.4 errors per document) on the test set compared to 7745 errors (138.3 errors per document) originally made by Tesseract on those same characters. For the 10 cases where SIFT_Align_Clean increased error, SIFT_Align_Clean made 356 character errors and Tesseract made 263 errors. Thus, overall, the error reductions achieved by SIFT_Align_Clean were much greater than the errors introduced.

SIFT_Align_Clean outperforms Sift_Align_Tess. Average error reduction for SIFT_Align_Clean is 34.1% compared to 9.5% for Sift_Align_Tess. Error reduction is calculated as $(TT - ST)/TD$ where TT is # Tesseract errors in the test set, ST is # SIFT_Align errors in the test set and TD is # Tesseract errors in the document. SIFT_Align_Clean also reduces the character error in more documents than does Sift_Align_Tess.

Our test cases only consider properly segmented characters which account for about half of all the errors in these documents. The error reduction for SIFT_Align_Clean over all characters (segmented properly or not) is 20.3%.

Our experiments have shown that, on two separate sets of documents, our conservative clean sets have very low error rates, meeting the theoretical bounds presented, and that by relaxing the criteria slightly, we can get significantly larger sets while maintaining a low error rate. We have shown that using these clean sets to build document-specific models can significantly reduce OCR errors, and that traditional confidence measures do not result in the same benefits.

7. Applications to Other Domains

We believe that our method of identifying subsets of results for which we can achieve a very high bound of being correct can also be applied to other domains outside of OCR.

15. There are two measures of word confidence in Tesseract, described in the Tesseract documentation as "rating" and "certainty". We use "certainty".

One such domain is speech recognition. Here, our consistency check would be over acoustic signals rather than a patch of pixels from a scanned document. This check would be similar to Algorithm 1 except that we now apply the check to a segment of speech signal. In this speech recognition context, the segment is now the "glyph" g and we want to check whether the label assigned to g is reliable by comparing g with other segments from the same recording that are most similar to g. Acoustic segment g should be given the same label (in our terminology, dominated by that label) as its most similar segments. We can then form equivalence classes of easily confusable phonemes, and perform consistency checks on segments of speech that have been labeled as a word with an empty pseudo-Hamming distance of 1. We could then follow a procedure similar to the proof presented in Section 4 to bound the probability that segments of speech that pass such a consistency check were incorrectly labeled.

By using this framework for speech recognition, we can potentially obtain the equivalent of clean lists: portions of speech for which we are very confident the initial labeling was correct. This may allow us to refine the speech recognition model to be specific to the recording, for instance, allowing for a model that is specific to a particular individual's accent.

The application of our idea to speech recognition may involve a slightly different set of difficulties than when applied to OCR. For instance, identifying word segmentations may be more difficult. However, when training speech recognition systems, we may have access to an additional source of information in the form of closed captions. We can model the closed captions as a noisy signal of the ground truth, independent of the speech recognizer. Taking a conservative estimate of the closed captioning error rate, we can use the closed captions to reduce our bound on the probability of error by requiring that the closed captioning match the labeling given by the speech recognition system.

In Lamel et al. (2002), audio with closed captions is used to generate additional labeled training data for a speech recognizer, by aligning the speech recognizer output to the closed captioning and accepting segments where the two agree. Given the large amount of closed captioning data available, this scenario is particularly amenable to our method of generating high precision training data (at some cost to recall). Additionally, using a consistency check approach as presented in Algorithm 1 can yield advantages over using an ad-hoc check such as directly accepting segments where the speech recognizer and closed captioning agree. For instance, we may find it necessary to throw out words where the two agree if the word has many nearby phonemic neighbors with which it may be confused, and thereby likely reduce the error rate of the labelings used as training data.

Another potential application for our bound analysis is in the area of information retrieval using query expansion. As mentioned earlier in Section 2.2.2, query expansion is a technique used in information retrieval to add terms to an initial query based on the highly ranked documents of the initial query. One issue when performing query expansion is that matching with false positives can quickly lead to errors due to drift in the query. For image retrieval, Chum et al. (2007) give a method of spatial verification to eliminate false matches. In this context, queries are objects in images and images are represented using a bag-of-visual-words.

Let the original query image be Q, and let the set of images returned initially by the search engine be $\mathbf{R}$. The goal is to identify returned images $\{R_i\}$ that we believe contain the same object as Q with very high confidence. We can then add these images $\{R_i\}$ to the query and repeat the search procedure with this expanded query set, ideally increasing the number of relevant images returned by the search engine.

To reduce the possibility of adding a false match R_i to the query set, Chum et al. (2007) apply a spatial verification procedure as follows. A feature point in R_i matching a feature point in Q

generates a hypothesized transformation that would put the object in R_i in correspondence with the object in Q. If this hypothesis leads to at least a certain number of matching feature points in R_i and Q (same visual word and location after transformation), then R_i is spatially verified and added to the query set.

We could estimate a bound on the probability of a false match passing spatial verification by assuming a feature in a random, non-matching returned image matches a feature in the query image Q with probability p. If we further assume that the probability of two features matching is independent of the other features in Q and returned image R_i, then the number of features in correspondence between the Q and non-matching result R_i is a binomial distribution with parameters n, the number of features in the query image, and p.

A result image R_i passes the spatial verification if, for at least one hypothesized transformation for putting Q in correspondence with R_i, at least 20 features are in correspondence. With the number of hypotheses being approximately 10^3, n also being approximately 10^3, and a conservatively high estimate of p as 10^{-3} (given a visual dictionary of size 10^6), we find that even with a requirement of just at least 12 features being in correspondence, the probability of a false match being spatially verified is less than $10^3 \cdot (1 - \sum_{i=0}^{11} \binom{n}{i} p^i (1-p)^{n-i}) < 10^{-6}$.

However, spatial verification will likely result in more errors than predicted by this analysis, due to the overly restrictive assumption that the probability of features matching in a result image R_i is independent of the other features. One potential method for removing this assumption is to analyze the error in terms of common substructures: features belonging to similar substructures are more likely to match, but the probability of one feature matching is independent of the features in different substructures of the same image. This analysis may suggest ways of improving the spatial verification, such as requiring that the matching features not be closely clustered in only one section of the image.

8. Conclusions and Future Work

In this paper, we advocate dealing with the problem of non-stationarity between the training and test distributions by identifying a subset of information whose interpretation we can be confident of, and using this information as test-specific training data. We have applied this approach to the problem of OCR, demonstrating that we can produce high-precision document-specific training data. Under modest assumptions, we show the error rate of this labeled data to be bounded by 0.002, and give empirical results consistent with this theoretical bound.

By combining this document-specific training data with simple appearance-based character recognition techniques, we are able to achieve significant reductions in average character error. We believe that further improvements can be achieved by using the clean lists in conjunction with more sophisticated models, such as document-specific language models, as suggested by Wick et al. (2007). In addition, while our work has taken the character segmentations produced by the initial OCR system as fixed, we believe that the clean lists can also be used to re-segment and fix the large percentage of initial errors that result from incorrect character segmentation.

Lastly, we also show potential applications to problems in other domains such as speech recognition and query expansion. While many of the techniques used in this work are specific to an OCR application, we believe that the principles are quite general, and that the search for more formal bounds on probabilities of error will lead toward a variety of new applications.

References

A. C. Berg, F. Grabler, and J. Malik. Parsing images of architectural scenes. In *International Conference on Computer Vision*, 2007.

T. M. Breuel. Character recognition by adaptive statistical similarity. In *International Conference on Document Analysis and Recognition*, 2003.

R. G. Casey. Text OCR by solving a cryptogram. In *International Conference on Pattern Recognition*, pages 349–351, 1986.

O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *International Conference on Computer Vision*, 2007.

J. Edwards and D. Forsyth. Searching for character models. In *Neural Information Processing Systems*, 2005.

T. K. Ho. Bootstrapping text recognition from stop words. In *International Conferece on Pattern Recognition*, pages 605–609, 1998.

T. K. Ho and G. Nagy. OCR with no shape training. In *International Conference on Pattern Recognition*, pages 27–30, 2000.

J. D. Hobby and T. K. Ho. Enhancing degraded document images via bitmap clustering and averaging. In *International Conference on Document Analysis and Recognition*, pages 394 – 400, 1997.

T. Hong and J. J. Hull. Visual inter-word relations and their use in OCR post-processing. In *International Conference on Document Analysis and Recognition*, pages 14–16, 1995a.

T. Hong and J. J. Hull. Improving OCR performance with word image equivalence. In *Symposium on Document Analysis and Information Retrieval*, pages 177–190, 1995b.

T. Hong and J. J. Hull. Character segmentation using visual inter-word constraints in a text page. In *Proceedings of the SPIE - The International Society for Optical Engineering*, pages 15–25, 1995c.

A. Kae and E. Learned-Miller. Learning on the fly: Font free approaches to difficult OCR problems. In *International Conference on Document Analysis and Recognition*, 2009.

A. Kae, G. B. Huang, and E. Learned-Miller. Bounding the probability of error for high precision recognition. Technical Report UM-CS-2009-031, University of Massachusetts Amherst, 2009.

A. Kae, G. B. Huang, C. Doersch, and E. Learned-Miller. Improving state-of-the-art OCR through high-precision document-specific modeling. In *Computer Vision and Pattern Recognition*, 2010.

O. Kolak. A generative probabilistic OCR model for NLP applications. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 55–62, 2003.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

K. Kukich. Techniques for automatically correcting words in text. *Association for Computing Machinery Computing Surveys*, 24(4):377–439, 1992. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/146370.146380.

L. Lamel, J. Gauvain, and G. Adda. Lightly supervised and unsupervised acoustic model training. *Computer Speech and Language*, 16(1):115–129, 2002.

D. Lee. Substitution deciphering based on HMMs with applications to compressed document processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1661–1666, 2002.

M. Leisink and B. Kappen. Bound propagation. *Journal of Artificial Intelligence Research*, 19(1): 139–154, 2003.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

G. Nagy. Efficient algorithms to decode substitution ciphers with applications to OCR. In *International Conference on Pattern Recognition*, pages 352–355, 1986.

G. Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.

M.-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation. In *British Machine Vision Conference*, 2007.

D. Ramanan. Learning to parse images of articulated bodies. In *Neural Information Processing Systems*, 2006.

B. Sapp, C. Jordan, and B. Taskar. Adaptive pose priors for pictorial structures. In *Computer Vision and Pattern Recognition*, 2010.

W. J. Scheirer, A Rocha, R. J. Micheals, and T. E. Boult. Meta-recognition: The theory and practice of recognition score analysis. *Pattern Analysis and Machine Intelligence*, 33(8):1689–1695, 2011.

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning*, 2004.

J. Weinman, E. Learned-Miller, and A. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1733–1746, Oct 2009.

M. Wick, M. Ross, and E. Learned-Miller. Context-sensitive error correction: Using topic models to improve OCR. In *International Conference on Document Analysis and Recognition*, 2007.