Learning on the Fly: Font-Free Approaches to Difficult OCR Problems

Andrew Kae and Erik Learned-Miller
Dept. of Computer Science
University of Massachusetts, Amherst MA
{akae,elm}@cs.umass.edu

Abstract

Despite ubiquitous claims that optical character recognition (OCR) is a "solved problem," many categories of documents continue to break modern OCR software such as documents with moderate degradation or unusual fonts. Many approaches rely on pre-computed or stored character models, but these are vulnerable to cases when the font of a particular document was not part of the training set, or when there is so much noise in a document that the font model becomes weak. To address these difficult cases, we present a form of iterative contextual modeling that learns character models directly from the document it is trying to recognize. We use these learned models both to segment the characters and to recognize them in an incremental, iterative process. We present results comparable to those of a commercial OCR system on a subset of characters from a difficult test document.

1. Introduction

Optical character recognition (OCR) has been a great success of computer vision and pattern recognition, but it is by no means "a solved problem." While there are many applications, such as internet search, that can benefit greatly from OCR in its current imperfect form, the goal of transcribing documents completely and accurately, under moderate degradation, variable fonts, interspersions of numerals and other common difficulties, is still far off.

In this work, we present an unsupervised OCR system which performs well on a real-world, degraded document, shown in Figure 1. **When a document is analyzed, the system has neither appearance models of any characters nor training data with examples of characters.** It is therefore a font-independent method of OCR. Instead of relying solely on the appearance of characters, font-independent OCR systems rely on the repetitions of similar symbols, coupled with statistics of a language, like English, to interpret a document.

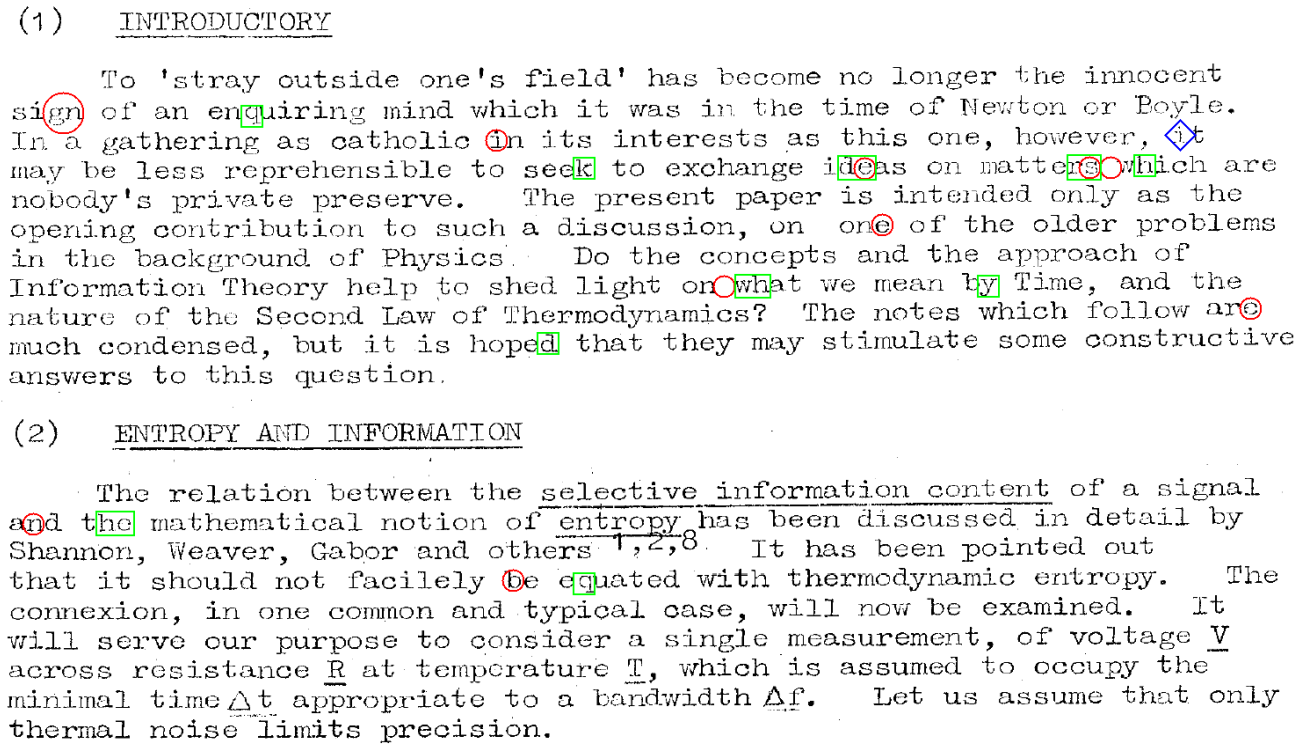
Our approach to OCR is a form of iterative contextual modeling, building a document-specific model by first recognizing the least ambiguous characters and then iteratively refining the model to recognize more difficult characters. Rather than rely on the appearance of characters relative to a model developed a priori, we compare characters to other characters in the same document to establish the equivalence of those characters. A language model then helps to determine the identity of the groups of similar characters by comparing word hypotheses to a frequency weighted English lexicon.

2. Background

Much early work in OCR used a rigid pipeline approach that used some approximation of the following sequence of steps: find text, segment the letters, recognize the letters, and then use a language model to correct errors. However, these models make strong assumptions that break down in challenging settings.

Systems that make hard decisions at each stage without benefit of later stages could only accumulate errors, except at the very end, in which language models were used to attempt to fix errors that had been made along the way. Such systems were brittle and were ultimately surpassed by systems that maintained degrees of uncertainty along the way, borrowing tools developed by the speech recognition community, such as hidden Markov models. In these systems, multiple hypotheses about both segmentations and character identities were maintained in a lattice framework, and a dynamic programming procedure was used to find the maximum likelihood interpretation according to a Markov probability model. Such systems today are at the heart of many OCR systems, and have been pushed quite far, as can be seen for example, in the work of Jacobs et al. [5].

One assumption of these systems is that the classifier used to evaluate characters has been trained on a font which is either equivalent, or highly similar to, the font or fonts which appear in the target document. Even if a modern OCR system has been trained with a very large number of



(1) INTRODUCTORY

To 'stray outside one's field' has become no longer the innocent sign of an enquiring mind which it was in the time of Newton or Boyle. In a gathering as catholic in its interests as this one, however, it may be less reprehensible to seek to exchange ideas on matters which are nobody's private preserve. The present paper is intended only as the opening contribution to such a discussion, on one of the older problems in the background of Physics. Do the concepts and the approach of Information Theory help to shed light on what we mean by Time, and the nature of the Second Law of Thermodynamics? The notes which follow are much condensed, but it is hoped that they may stimulate some constructive answers to this question.

(2) ENTROPY AND INFORMATION

The relation between the selective information content of a signal and the mathematical notion of entropy has been discussed in detail by Shannon, Weaver, Gabor and others [1,2,8]. It has been pointed out that it should not facilely be equated with thermodynamic entropy. The connexion, in one common and typical case, will now be examined. It will serve our purpose to consider a single measurement, of voltage V across resistance R at temperature T, which is assumed to occupy the minimal time Δt appropriate to a bandwidth Δf. Let us assume that only thermal noise limits precision.

Figure 1. Document used in experiments. Errors made by our system are shown in boxes. Errors made by OmniPage are shown in circles. Errors made by both systems are shown in diamonds. A blank circle indicates an extra character was added in the OCR output.

fonts, document noise can significantly alter the appearance of such fonts, making them a poor match to the stored fonts.

When the apperance model is poor, it may seem that an OCR system is lost, but it is still possible to recognize documents, even when there is no appearance model at all. Previous work has shown that if the characters in a document can be clustered by appearance (which does not require an appearance model for each character class), then even if the identity of each character is initially unknown, it can be inferred simply by leveraging the statistics of the occurrence of each character [4, 6, 7, 9]. Huang et al. [11] give the example of a word encoded with random Greek letters α β γ γ β γ γ β δ δ β which matches only to the word *Mississippi* using an English dictionary. This illustrates the idea that *repetitions of appearance*, rather than models of appearance, can be enough to infer the identity of letters in a document. Such methods are sometimes referred to as *ciphering* or *cryptogram decoding* methods.

Treating OCR as a cryptogram decoding problem dates back at least to papers by Nagy [14] and Casey [6] in 1986. In [7], Ho and Nagy develop an unsupervised OCR system that performs character clustering followed by lexicon-based decoding. In [12], Lee uses hidden Markov models to decode substitution ciphers based on character clusters. Breuel [4] also presented a probabilistic method of clustering characters based on the similarity of their appearance.

These previous approaches to font-independent OCR have shown intriguing results, but have been limited by two critical factors. They all assume that characters can be segmented accurately as a first step, which is known to be a very difficult problem. Second, with the exception of the work by Ho and Nagy [7], they assume that all characters can be grouped into pure clusters, i.e. clusters that contain only a single type of character. However, these assumptions are too strong to apply to anything but very clean documents.

3. Learning on the Fly

Contributions. In this paper, we build on many of the ideas of previous papers. The work most similar to our own is probably that of Ho and Nagy [7] which also incorporates language statistics into a document-specific model. We introduce the following innovations.

- Instead of segmenting characters first, we interleave

segmentation with character recognition in an iterative process.

- Our approach first recognizes easier, unambiguous characters and then the language model uses these partial recognitions to build more context from which to evaluate more difficult characters.

- There is an iterative joint inference between language and appearance models. In each iteration, the appearance classifier fixes probable mistakes made by the language model, which improves the language model for the next iteration.

Terminology. A *blob* refers to any group of characters that has not yet been segmented. Initially, every word in the document is considered a blob.

A *glyph* represents a rectangular portion of an image, which is likely to be a single character, but may represent a portion of a character, multiple characters, or a stray mark. A glyph is a blob which is being considered as a character candidate.

A *glyph set* is a collection of glyphs that are considered to be the same character. This can be thought of as a cluster, but we do not use the term cluster since the glyph sets are not obtained through a clustering process.

Recognition is the assignment of a glyph set to a character class, like "e".

Matching is the process of comparing a glyph to a set of blobs in order to find more instances of the same glyph (typically using normalized cross correlation). If a glyph is matched to a portion of a blob, this will segment the blob into smaller blobs. This approach to segmentation is similar to Hong et al [10].

A *redacted string* is a string representing a word in the document and is a mixture of assigned characters, unsegmented blobs and a special placemarker θ representing the glyph we want to recognize. The unsegmented blob has an approximate length associated with it. For example, the word "apple" might have the redacted string "a$\{3\}\theta$" if the "a" glyph was already assigned, θ is the glyph we want to label, and there is a blob approximately 3 characters long in between the "a" and θ. By matching a redacted string against our lexicon and obtaining a list of words that are consistent with the redacted string, we can assess the probability of θ being a particular character.

A *dominant font* is a font (comprising both a style such as Helvetica and a specific point size) that represents more than 50% of the characters in a document. If no single font represents more than 50% of the characters in the document, we say that there is no dominant font.

Assumptions. The system we have built so far is not a commercial grade OCR system. It is intended to illustrate the feasibility of the ideas presented here. However, the example we use in our experiment is from a challenging real-world document [13], so it is not artificial data. Nevertheless, we have made a number of assumptions to get off the ground, which we hope to relax in future research.

We assume the language of the document is English and the distribution of words is similar to the one in our corpus. We assume the text we want to recognize is in the dominant font and we can find a sample of the *word* "a". This unusual assumption helps us to bootstrap our segmentation process. A somewhat similar assumption about the frequency of stop words has been made by Ho et al [8].

Finally, we do not address the important problems of page layout, line finding, and the segmentation of words within lines, as these are handled sufficiently for our purposes by the methods of others for the time being. The segmentation of letters within words, however, is a key focus of our work.

Method. We now present our unsupervised, document-specific method:

1. First, we find the word "a" in the document, which obviates character segmentation since the word consists of a single character. To do this, sort all words by their width in pixels. For the N shortest words (we choose N to be 15), match each word by performing a normalized correlation of the word's rectangular bounding box to every position in the document (searching only along the baselines of text lines). A "match" is found if a specific threshold of the normalized correlation is reached. For each of these N words, assume that the word is indeed an "a", and use the width of the word to calculate about how many total characters there are in the document. Then, knowing that the lower case letter "a" accounts for about 7% of all letters, pick the word that most closely matches this percentage.

2. Match the "a" found in the previous step to all blobs in the document, building our first glyph set. When matching, our method strongly favors high precision over high recall.

3. Each candidate "a" in the glyph set (some of the elements in the set may be erroneously included) is used as a new delimiter to *shatter* the current blobs into smaller blobs.

4. From the remaining blobs, select a new blob θ as the next glyph for matching. Unlike the previous steps, we do not know, a priori, the identity of θ. We match each potential blob to form a glyph set. After several glyph sets have been created, we use the language model (see next section) to choose and identify the glyph set in which it has greatest confidence. We measure this by

picking the glyph set with the largest difference in likelihood between its most confident and next most confident character assignment.

5. Add the glyph set of θ and its assigned character class found in the previous step to the model.

6. Reclassify all glyphs in every labeled glyph set by comparing each glyph to the mean glyph of every labeled glyph set and move it to the glyph set it matches highest to (as measured by normalized cross correlation). The mean glyph is an average of the glyphs in a glyph set. This allows us to recover from early errors in glyph set creation.

7. Steps 4-6 are repeated until none of the remaining blobs match to any part of the remaining document, i.e. no glyph set with more than a single glyph can be produced. Each blob is then recognized by comparing it to its 5 nearest neighbors (as measured by normalized cross correlation) across all glyph sets, and using the majority vote as the assigned character class.

Language Model. We use corpus statistics to calculate the likelihood of observing a set of redacted strings. We use 10 texts from the Project Gutenberg corpus [1] to provide statistics of the English language. There were a total of 1,525,460 words in the corpus.

In the description below, θ is the glyph set to label, R is the set of redacted strings and V is the set of recall rates for previously assigned characters. We assume that the label of θ comes from some alphabet (in our case, we are assuming all ASCII characters). It is necessary to track recall rates because we may consider labeling a new glyph set with a character already assigned to a previous glyph set.

For example, if we observed the letter "a" with 90% estimated recall (so about 90% of the true "a"s have been found), it is unlikely that a new, large glyph set is really an "a". We approximate the expected number of "a"s in a document by multiplying the unigram frequency of "a" by the approximate total number of characters in the document. We approximate the total number of characters by dividing the combined width of all blobs by the width of the "a" glyph. We then compute recall by dividing the observed number of "a"s by the expected number of "a"s.

In a set of redacted strings R, each R_i may consist of previously assigned characters, unsegmented blobs, and a placemarker θ for a new glyph set we want to identify. Our goal is to find the most likely character class g given the redacted strings R and recall rates V.

$$P(\theta = g|R,V) = \frac{P(R|\theta = g,V) \cdot P(\theta = g|V)}{P(R|V)}$$

using Bayes Rule. Since $P(R|V)$ does not change for different g, we ignore it. We treat $P(\theta = g|V)$ as the unigram probability of g. We consider each redacted string R_i independently to get

$$P(R|\theta = g,V) = \prod_i P(R_i|\theta = g,V).$$

Our generative model assumes that each R_i has a 95% chance of being drawn from our word corpus and a 5% chance of being generated by a random character process:

$$P(R_i|\theta = g,V) = 0.95 \cdot (\sum_{m \in matches} P(m) \cdot \prod_{c \in chars(m)} P(r(c)|V,R_i) \cdot \prod_{b \in blobs(R_i)} P(b|R_i)) + 0.05 \cdot (\prod_{c \in chars(R_i)} P(c) \cdot \prod_{c \in chars(R_i)} P(c|V)).$$

For the probability of R_i being drawn from a corpus word, we sum over all words m that match the redaction and calculate $P(m)$, the probability of observing the matching word m in our corpus.

Next, for each character c in m, we compute $P(r(c)|V,R_i)$ using a Bernoulli distribution based on the observed recall rates. $P(r(c)|V,R_i)$ = recall of c, if c is found in V, and (1-recall of c) if not. For example, if the redaction is "{1}ye" and m = "eye", $P(r(\text{"e"})|V,R_i)$ = (1-recall of "e") since we missed it, then $P(r(\text{"y"})|V,R_i)$ = recall of "y", and lastly $P(r(\text{"e"})|V,R_i)$ = recall of "e". If the recall for "e" is high, we don't expect to observe many more "e"s and so (1-recall of "e") will be low.

We use the length information in a redacted string to penalize matches to words with significantly different lengths. For example, if the redaction is "{1}at", this matches to a word like "cat" but also to a word like "what". If the blob length is off by 1, we set $P(b|R_i)$ =20%, if it the same length, $P(b|R_i) = 60\%$, and $P(b|R_i) = 0\%$ otherwise.

Because we need to account for the possibility that a redaction may not match to any corpus word, we reserve a probability mass of 5% for unseen cases. We compute this portion by examining each character c in the redaction (ignoring the blobs) and multiply the unigram probability of c and the recall rate $P(c|V)$.

When we want to recognize a glyph set θ, we consider $\theta = g$, where g can be any ASCII character. We then run the calculations described above and pick the argmax of $P(\theta = g|R,V)$.

4. Experiments

We evaluated our system on a portion of an old IEEE paper [13] shown in Figure 1. We attempted to recognize all word blobs in the document but for evaluation, we currently only consider words entirely in lowercase, which we call *Level 0* words. This excludes any words containing uppercase letters, digits or special characters. These characters occur less frequently than most lowercase letters and so our

Table 1. Results

Method	Word Accuracy	Character Accuracy
Ours	164/175 = 0.937	0.981
Omnipage	164/175 = 0.937	0.979
Tesseract	172/175 = 0.983	0.996

Table 2. Sample Errors

Method	Errors
Ours	it → t, the → t, equated → eluated
Omnipage	it → It, and → arid, be → he

language model doesn't have enough leverage to accurately classify these glyph sets.

We ran experiments on a Core2Duo 2.66GHz machine with 2GB of RAM. The entire process took about 12 hours to run with most computation time spent on redacted string calculations. When we began our research, Omnipage 15 [2] had the best performance of several systems on our test document, and our goal was to achieve comparable or better performance on the dominant font for this document. Just before submission, we tried a new release (2.03) of Tesseract [3], an open-source OCR system, and its results had improved substantially, beating both our own results and those of Omnipage 15. Despite coming in behind Tesseract, we feel our results are still of significant interest, since we produced competitive results with no font models at all as shown in Tables 1 and 2.

Following the conventions in [15], we define character accuracy as $n - \#errors/n$, where n is the total number of correct characters and # errors is the number of edits needed to correct the OCR output text. Word accuracy is defined to be the percentage of correctly recognized words.

In Table 2, we list some of the representative errors made by each system on the text in Figure 1. Our approach and OmniPage incorrectly classified "it" as "t" and "It" respectively. The "i" glyph in the diamond in Figure 1 is highly degraded and was not recognized as a glyph candidate by our system. Our system misrecognized "the" as "t", illustrating a weakness in our matching. Since no glyph matched to the "he" portion, it was left unrecognized. Our approach also misrecognized "equated" as "eluated". This is because the "q" glyph is so infrequent, there isn't enough context for correct labeling. Our approach is strongest when there are many examples of a glyph.

5. Conclusion and Future Work

We have shown that it is possible to achieve favorable results on a real, noisy document using only a corpus of text and without using any character models or training data. In the future, we plan on testing our approach over a much larger set of test documents and relaxing constraints to allow for full recognition of all character types.

One significant drawback of our approach is the long time necessary to compute the most likely character recognition of a set of redacted strings. For our approach to be viable and used in production, this computation time must be reduced significantly.

Our approach assumes that test documents conform to the English word statistics found in our corpus. If we use a topic model [16], we can determine that certain words are more likely to be observed than others. For example, if a document is about cars, then we are more likely to observe a word like "torque" than "tongue" which may help if the images for "rq" and "ng" appear very similar.

References

[1] `http://www.gutenberg.org/`.

[2] `http://www.nuance.com/omnipage/`.

[3] `http://code.google.com/p/tesseract-ocr/`.

[4] T. Breuel. Classification by probabilistic clustering. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2001.

[5] P. V. C. Jacobs, P. Y. Simard and J. Rinker. Text recognition of low-resolution document images. In *ICDAR*, pages 695–699, 2005.

[6] R. Casey. Text OCR by solving a cryptogram. In *International Conference on Pattern Recognition*, 1986.

[7] T. Ho and G. Nagy. OCR with no shape training. In *International Conference on Pattern Recognition*, 2000.

[8] T. K. Ho. Bootstrapping text recognition from stop words. In *Procs. ICPR-14*, pages 605–609, 1998.

[9] J. Hobby and T. Ho. Enhancing degraded document images via bitmap clustering and averaging. In *International Conference on Document Analysis and Recognition*, 1997.

[10] T. Hong and J. Hull. Visual inter-word relations and their use in character segmentation. In *SPIE*, 1995.

[11] G. Huang, E. Learned-Miller, and A. McCallum. Cryptogram decoding for optical character recognition. In *International Conference on Document Analysis and Recognition*, 2007.

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

[13] D. MacKay. Entropy, time and information (introduction to discussion). *Information Theory, Transaction of the IRE Professional Group*, 1(1):162–165, 1953.

[14] G. Nagy. Efficient algorithms to decode substitution ciphers with applications to OCR. In *International Conference on Pattern Recognition*, 1986.

[15] S. V. Rice, S. V. Rice, F. R. Jenkins, F. R. Jenkins, T. A. Nartker, and T. A. Nartker. The fifth annual test of ocr accuracy. Technical report, 1996.

[16] M. Steyvers and T. Griffiths. Probabilistic topic models. In T. Landauer, D. McNamara, S. Dennis, and W. Kintsch, editors, *Latent Semantic Analysis: A Road to Meaning*. Laurence Erlbaum, 2006. In press.